

Report from the Southeast Region Workshop*
on
Integrative Computing Education and
Research (ICER): Preparing IT Graduates
for 2010 and Beyond

October 27-28, 2005
Dallas, Texas

Prepared By:

Dr. Oscar Garcia
Dr. Lillian Cassel
Dr. Kathleen Swigger

*** This workshop was sponsored by the National Science Foundation and the College of Engineering at the University of North Texas under NSF Award 0547299.**

Participant List

Antonio, John	University of Oklahoma
Bayoumi, Magdy	University of Southwestern Louisiana
Bridges, Susan	Mississippi State University
Cassel, Lillian *	Villanova University
Castaneda, Sheila	Clarke College
Cross, James H. II	Auburn University
Doran, Mike	University of South Alabama
Elmaghraby, Adel S.	University of Louisville
Fishwick, Paul	University of Florida
Furuta, Richard	Texas A & M University
Garcia, Oscar *	University of North Texas
Gupta, Sandeep K.S.	Arizona State University
Guzdial, Mark	Georgia Institute of Technology
Hadzikadic, Mirsad	University of North Carolina – Charlotte
Huhns, Michael N.	University of South Carolina
Jovanovic, Vladan	Georgia Southern University
Lawhead, Pam	University of Mississippi
Peterson, Greg	University of Tennessee – Knoxville
Pfeiffer, Phil	East Tennessee State University
Reed, Dan	University of North Carolina
Reichgelt, Johannes "Han"	Georgia Southern University
Rodriquez-Rodriquez, Domingo	University of Puerto Rico Mayagüez
Roszbach, Uwe **	University of North Texas
Rutherford, Rebecca	Southern Polytechnic State University
Solano, Judy	University of North Florida
Srimani, Pradip K.	Clemson University
Swigger, Kathy *	University of North Texas
Thompson, Craig W.	University of Arkansas
Varanasi, Murali	University of North Texas
Vasquez, Ramon	University of Puerto Rico Mayagüez
Wardle, Caroline	National Science Foundation
Willis, Cheryl	University of Houston
Zhao, Wei	National Science Foundation

* **Workshop Hosts**

** **Computer Support**

Table of Contents

Executive Summary	4
NSF (CISE) Program Executive Summary	5
Workshop Report	
Introduction	6
Summary of Discussions	7
Conclusions & Suggestions	10
Appendix A: Agenda	11
Appendix B: Group Reports	
Group 1 Report & Presentation	13
Group 2 Report & Presentation	18
Group 3 Report & Presentation	24
Group 4 Report & Presentation	30
Appendix C: Individual Reports	
Antonio, John	37
Bridges, Susan	38
Castaneda, Sheila	40
Cross, James H. II	42
Doran, Mike	44
Elmaghraby, Adel S.	46
Furuta, Richard	48
Fishwick, Paul	50
Gupta, Sandeep K.S.....	52
Guzdial, Mark	55
Hadzikadic, Mirsad	57
Huhns, Michael N.	58
Jovanovic, Vladan	60
Lawhead, Pam	62
Peterson, Greg.....	64
Pfeiffer, Phil	66
Reichgelt, Johannes "Han"	68
Rutherford, Rebecca	70
Solano, Judy	72
Srimani, Pradip K.....	74
Thompson, Craig W.....	76
Append D: Invited Speaker Presentations (separate document)	
Presentation slides for Shelia Castaneda	79
Presentation slides for Dan Reed	101
Presentation slides for Murali Varanasi	119

Integrative Computing Education and Research (ICER): Preparing IT Graduates for 2010 and Beyond

October 27-28, 2005

Executive Summary

The popularity of computer science at US undergraduate institutions has declined dramatically since 2001, despite projections of large increases in demand for IT workers, with enrollment levels dropping to lows not seen since the early 1970s. Educators have attributed the decline to factors that include changes in the USA workforce, the “geek” image, outsourcing, the dot-com meltdown, curriculum malaise, and a movement toward courses that apply rather than program computers. Academic discussions of these and other problems have raised questions about how well current computing programs are meeting the needs of their constituents and whether they can address the growing concern about keeping America competitive.

To obtain community input on these issues, an NSF-sponsored workshop was held on October 27-28, 2005, at the American Airlines Conference Center in Dallas, Texas. Twenty-seven participants from southern universities, colleges, and foundations with strong interests in computing and education met to assess the future needs of computing and information education and career development and develop recommendations to help assure the availability of future generations of highly-trained professions in the field of computing.

The principle recommendations included:

- identifying a set of principles common to contemporary specializations in computing (CS/IS/IT/SE);
- publicizing the nature of computing and the need for computing professionals, through public service announcements, websites, and joint programs between colleges and K-12 instructors;
- improving training for primary and secondary teachers in math and computing;
- modernizing content at *all* levels of the curriculum, through the presentation of newer technologies (e.g., robotics, wireless), applications of computing (e.g., bioinformatics, nanotechnology), and perspectives on computing’s role in the global economy;
- modernizing instruction, using new instructional media (e.g., PDAs, chat rooms, Wikis) and methods (e.g., visualization, reasoning about problem solving);
- expanding the curriculum’s scope, by providing research experiences for undergraduates that stress multidisciplinary teams, and offering team-taught courses in ‘hot cross-disciplinary topics’ like bioinformatics and nanotechnology;
- expanding the focus on research, by offering lower-level courses on innovative technologies, and studying how engineering research centers can be used to drive change;
- reviewing the role of math in the computing curriculum;
- making the curriculum more open to students without traditional math/science backgrounds—e.g., by offering a cross-disciplinary degree in applied computing;
- creating modules that facilitate the incorporation of new material into existing curricula;
- rewarding computing faculty for participation in educationally related activities;
- clarifying the role of accreditation in promoting curricular content and quality; and
- offering workshops that promote faculty retraining and curriculum revision, and that focus attention on the perspectives and needs of today’s students.

Executive Summary

"Integrative Computing Education & Research (ICER): Preparing IT Graduates for 2010 and Beyond"

Introduction

NSF's CISE Directorate is proactive in identifying and addressing issues related to undergraduate computing education in the nation. In this context, CISE is taking a five to ten-year view of the field of computing education and the impacts of trends such as: decreasing enrollments in academic computing programs, needs of the USA workforce, national demographics, shifts in global competitiveness, movement towards multidisciplinary domains of knowledge in computing applications, the integrative nature of the field of computing, and future grand challenges that may face the field of computing.

Stimuli for Strategic Planning

The driving forces for ICER planning are domestic and global events and trends that impact on the nation's competitiveness and the maintenance of its intellectual resource base in computing including:

- The intellectual content of the field of computing has changed radically. It affects other fields, is affected by other fields, and involves understanding many more complex interactions and integration than in the past. For the most part, computing curricula do not address an integrative view of the field nor have curricula kept pace with industry needs and challenges posed by ever expanding and increasingly complex applications.
- There is no uniform agreement about what constitutes the core of the computing field or how to produce graduates who are intellectually agile in a dynamically changing discipline. Typically, multiple IT programs exist on campus. Cross campus coordination and integration of these programs will improve the efficiency and effectiveness of education and research for every one.
- Graduates of computing programs typically are lacking a systems approach toward solving problems. They are not adept at dealing with the scale-up challenges associated with complex systems of the type they will encounter as practitioners.
- The dwindling pipeline of high school graduates majoring in computing and the underrepresentation of women and other minorities enrolled in computing programs or working as practitioners persists. In the past, international students compensated for the dwindling pipeline, studying both at the undergraduate and graduate levels and most often remaining in the USA workforce after graduation. However, with restrictions on visas, the USA lost this important source of students and practitioners.
- National IT competitiveness is threatened by global economies in a number of ways (e.g., offshoring/outsourcing, emergence of new information-based centers such as in the mid-east, and government supported software development industries such as in Ireland, Israel, and Poland).
- Security has become one of the nation's most pressing immediate needs.

Vision

To foster integrative computing education, CISE envisions a series of activities that will involve major stakeholder groups in the USA. The proposed stakeholders will be selected from groups such as: computing faculty, academic administrators, representatives of professional computing societies and trade organizations, government policy makers and funding organizations, recognized nation leaders and futurists in the field of computing, and representatives of national research and industrial laboratories. The activities will address:

- Campus-wide integration of IT education and research,
- Designing computing curricula that reflect the integrative nature of the field,

The outcomes will be long-term, high impact, and potentially high-risk, strategies to catalyze the transformation of university computing education throughout the nation.

Introduction: Workshop Format

In order to prepare for the workshop, participants wrote two-page white papers on the workshop's objectives. Each participant was asked to address the following five questions in this paper:

1. *Preparing undergraduates for computing careers:* What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)?
2. *Transforming the educational experience:* What might the community do to address the challenges you identified above?
3. *Models for transforming computing education:* What might an ideal undergraduate model for computing education look like in five years?
4. *Inhibitors and strategies:* Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA?
5. *Who might participate:* What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation? What is the role of government in this process? Professional societies? Universities and faculty? Others?

These papers were posted on a special website where attendees could view and comment on content. Additional comments were made during the first session of the workshop, which included oral presentations of the papers by each participant.

Supplementary ideas were offered to participants through presentations by three major speakers. Sheila Castañeda, showed Diana Oblinger's video presentation on the learning styles of the net generation, and commented on Dr. Oblinger's findings. The second, Dr. Dan Reed, discussed the multidisciplinary aspects of the computing field. The third, Dr. Murali Varanasi, described an innovative course at the University of North Texas that teaches students how to 'learn-to-learn.'

To ensure active interaction, opportunities for participation, and adequate consideration of the workshop's themes, the participants were divided into four groups, one for each of the workshop's four major themes: preparing undergraduates for computing careers, transforming the educational experience, models for transforming education, and inhibitors and strategies. All four groups were also asked to incorporate the last theme—who might participate—into their discussions. Each group maintained focus on its primary question, in spite of the overlap in expertise amongst the groups and the commonalities amongst the themes. Each discussion concluded with a summary of findings and recommendations.

Following group discussions, participants presented their views and recommendations in a plenary session. As a final exercise during the plenary session, each attendee stated the recommendations they considered to be the highest priorities for research, training, education, and career development.

Summary of Discussions

1. *Preparing undergraduates for computing careers:* What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)?
 - The new generation of students is vastly different from the past. They
 - are more active and visual learners,
 - have shorter attention spans and like to work in groups,
 - come to college exposed to many different kinds of media,
 - are more ‘connected’ to their peers, and
 - are more interested in applications that can solve specific problems or benefit society (See *I.A, II.A, III.C*)¹.
 - Several challenges were identified that relate to how well undergraduate students are prepared academically for computing curricula. Many freshmen who lack the prerequisite skills in math and science avoid taking computing courses. Others flunk out their first year. Some of their math and science phobia is caused by the poor introduction to mathematics and science by grade-school and middle-school teachers who lack sufficient preparation in these fields. In addition to the weak preparation of some elementary and middle school teachers of mathematics and science, the lack of interest in these topics among many teachers represents a disincentive to children who might have an interest in pursuing mathematical and scientific careers. (See *I.A; III.C, IV.6*)
 - Core curricula in some computing departments have not been updated to take advantage of the dynamic approaches to problem-solving and the new advances in computer science. While most students are comfortable with using tools like PDAs, chat rooms, and Wikis, these new technologies are rarely incorporated into computing courses (See *I.B.5, II.B, III.A.3*).
 - The more interesting computer application classes such as bioinformatics, robotics, and wireless applications are not usually taught in the early stages of undergraduate education, so students lose interest and drop out before they appreciate the vast array of career opportunities available to them (See *I.B.4*).
 - Computing curricula usually do not address the new requirements that global economy demands, especially the kinds of issues that are directed toward project development with partners who may be overseas (See *II.A, Reed*).
 - Research-oriented computing departments generally do not reward (i.e., tenure or promote) faculty for participation in educationally related activities such as curriculum reform. This serves as a disincentive for faculty at these institutions to participate in such activities (See *II.B & E, IV.A.6*).

¹ Notation refers to items in group reports in Appendix C – eg., I is group 1, II is group 2, etc.

- There is serious concern about losing the best and brightest students, including women and minorities, to other science disciplines. We appear to favor the expansion of computing curricula, but do not yet understand how to handle totally new areas (e.g., bioinformatics, nanotechnologies) (See *II.A, Castaneda, Reed, Varanasi*).
 - Some attendees expressed concern that definitions of computing related programs approved by accrediting agencies are narrow and restrictive. Interestingly, others suggested that current requirements lack rigor. Clarification of the appropriate role of accreditation and the ways in which accreditation criteria reflect and enforce a community view of meaningful and appropriate post secondary programs is needed (See *II.A & B, IV.4*).
2. *Transforming the educational experience*: What might the community do to overcome the challenges (cited above).
- Develop middle and high school programs that expand students' perspectives so that they might consider careers in computing. Involve teachers and counselors in activities that deal with computing. Ensure that grade-school, middle-school, and high-school teachers have sufficient mathematical and scientific background to enable them to instill a love of mathematics and science in their students (See *III.E, IV.A.6*).
 - Develop materials (online or otherwise) and public announcements that introduce students to career opportunities in computing (See *IV.A.5*).
 - Provide research experiences for undergraduate majors and stress multi-disciplinary teams (See *I.B.6, II.B*).
 - Create a program whereby Computer Science and Engineering educators become 'recertified' every five or six years by learning about new educational techniques and best practices. This should include learning about teaching methodologies that best reach the students who are entering college today (reported in large-group discussion).
 - Consider team teaching courses with researchers and educators in the 'hot-new-areas' such as bioinformatics and nanotechnology (See *I.B.5*).
3. *Models for transforming computing education*: What might an ideal undergraduate model for computing education look like in five years?
- Schools and Colleges should re-visit the idea of establishing a common core for all undergraduate majors. Conversations about this subject ranged from introducing courses in the liberal arts area to re-examining basic computing principles (See *I.B.3, III.B, IV.A.2*).
 - The computing community should develop curriculum "modules" that are shorter than courses and can be taught in less than a semester. The idea is to serve the needs of

different communities by creating different threads that connect a series of similar content or career modules (See *I.B.4, II.D*).

- There is a need for careful review of the mathematics requirements of computing related programs. Mathematics is important, but there is reason to question whether the mathematics that is generally required in existing programs is the right preparation. (See *II.D*)
 - Special research courses should be established that introduce students to innovative technology in their first or second year (See *I.B.6*)
 - The discipline might want to study the structure of Engineering Research Centers and use them as a possible model for innovative changes (reported in large-group discussion).
 - Departments should consider making their curriculum more open to students who lack a traditional math/science background and develop niches that allow these people to get degrees in computer science (See *II. B, III.A*).
 - Workshops in the use of current accreditation criteria to develop creative and innovative programs in computing should be made widely available. These could form the basis for a general approach to faculty retraining, curriculum revision, and attention to the needs of today's students (See *IV.4, Varanasi*)
4. *Inhibitors and strategies*: Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA?
- In the early years, computer science programs were more open to students as well as faculty with nontraditional backgrounds or eclectic interests. Over the years, computer science has gotten more narrowly focused and exclusive (See *II.D, Reed*)
 - New forms of computing related programs are emerging. These need to be monitored and documented with special attention to the goals of various types of programs. Identification of holes in the fabric of overall computing and information-related career preparation needs to be identified and filled. One suggestion from this workshop was to develop a degree in Applied Computing (See *I.B.3, reported in large-group discussion*).
 - There seems to be a lack of creativity and fresh ideas that can jump start reform. Computing educators need to be more future-oriented and incorporate new technology and educational delivery systems. Opportunities for a group of departments to work together to explore ideas, implement prototypes and experiment to determine what really works over a broad spectrum of environments are very rare. Some possibilities for change impact the entire institution, which raises another level of inhibitor (See *I.C, Castañeda, Reed*).

- Many faculty feel out of synch with students of the current generation. Instruction of faculty in the facts about their students' perspectives on the world should be incorporated into workshops that support innovation in curriculum design and teaching methods (See *I.B.6; IV.A.5, Castañeda*).
- There is generally no support (or tenure) for people who want to develop interdisciplinary programs. Yet these are the types of programs that seem to attract this generation of students (See *II.E*).
- Women and minorities are not attracted to the computer science field. If the discipline fails to reverse this trend, then the number of people entering computer science will continue to fall (See *I.A, II.A. Castañeda*).

Conclusions and Suggestions

The most important challenge identified in the workshop is to *convert the findings into viable and effective programs that can attract the next generation of computer science students* without compromising the academic quality of the programs. The workshop attempted to provide a detailed assessment of the challenges facing computing educators and suggested actions that can be taken to make careers in computing more attractive. This report serves as an overview of those discussions and conclusions. While there were some differences in how individuals perceived specific problems, there was noticeable agreement in the need to reform and change directions in computing education.

Appendix A. Agenda

Southeast Region Invitational NSF Workshop on “Integrative Computing Education and Research (ICER): Preparing IT Graduates for 2010 and Beyond”

Dallas, Texas

October 27-28, 2005

4501 Highway 360 South

Fort Worth, Texas

Website for Conference: <http://www.eng.unt.edu/highlights.htm>

Website for Papers: <https://webctvista.unt.edu/webct/>

Meals – Black Hawke Ballroom 1

Meeting Room – M106 Mercury

Wednesday, October 26

6:00 - 9:00 PM Registration for Workshop (in **Library off** the lobby)

Thursday, October 27

7:00 – 8:30 AM Breakfast in the Black Hawke 1

8:30 – 9:00 AM Mercury M106 -- Registration for late arrivals

9:00 – 9:05 AM Mercury M106 -- Welcome to the workshop (UNT Founding Dean Oscar Garcia)

9:05 – 9:20 AM Welcome and overview (Dr. Wei Zhao, Division Director CNS, NSF)

9:20 – 10:20 AM Each attendee will give a **5 minute** (max) presentation of their paper presenting their views on topics such as:

1. *Preparing undergraduates for computing careers*: What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)?
2. *Transforming the educational experience*: What might the community do to address the challenges that you identified above?
3. *Models for transforming computing education*: What might an ideal undergraduate model for computing education look like in five years?
4. *Inhibitors and strategies*: Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of university computing education throughout the nation?
 - *Who might participate*: What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation? What is the role of government in this process? Professional societies? Universities and faculty? Others?

10:20 – 10:30 AM Break

10:30 – 11:30 AM Continuation of participant presentations

11:30 - 12:30 PM Lunch

- 12:45 -1:45 PM Presentation: “The Net Generation” video by Diana Oblinger with commentary and critique by Sheila Castaneda, Clarke College
- 1:45 – 2:00 PM Break
- 2:00 – 3:30 PM Discussion by all participants on the topics presented during the morning session; formation of 4 small groups to address each of the specific areas cited above (1-4) with the last one answered by all.
- 3:30 – 4:00 PM Break
- 4:00 – 5:00 PM Continued discussion by small groups on the issues that were identified and compilation of ways to address specific problems. Each group will designate a speaker for the group and a scribe to takes notes of the group’s discussion. These notes will serve as the basis for the group’s report, to be presented the following morning. Participants should begin writing reports.
- 7:00 – 9:00 PM Dinner. **Americana 2** - After-dinner Presentation: “Computing: The 21st Century Liberal Arts Education” by Dr. Dan Reed, University of North Carolina at Chapel Hill.

October 27

- 7:00 – 8:30 AM Breakfast
- 8:30 – 11:30 AM Continued small group discussions review of final report.
- 11:30 – 1:30 PM Lunch followed by Presentation: “L2L: Learning to Learn” by Dr. Murali Varansi, University of North Texas
- 1:30 – 1:45 PM Break
- 1:45 – 3:00 PM Small group presentations (10-15 minutes) to all participants followed by general summary and conclusions.
- 3:00 PM Dismissal of workshop participants with a thank you to all who attended.

Appendix B. Group Reports

I. Group 1 Report

Question 1 – Preparing Undergraduates for Computing Careers: *What are the challenges that we face in our role (as an educator, employer, administrator, leader, other) in preparing students for computer careers?*

A. Incoming Students

The **diversity of backgrounds and goals among incoming students** is a particular challenge. Students come to us unprepared for the rigors of a university education (including mathematics, writing, thinking skills) and with unrealistic expectations of what the computing field encompasses. Their computing experiences in high school may be limited or non-existent (so they don't know what to expect), or may be such a narrow viewpoint of computer science that they do not want to pursue it as a career. Women, in particular, are turned off by the office training, game-playing, hacking, vendor-specific training- mentality that is often portrayed as computer science.

Beginning courses need to educate and engage a wide variety of students. Keeping students interested (and not dropping out of our discipline) when they have a range of programming/computer experience from considerable to none, is a daunting task. A related problem is the influx of community college students two years later who have a wide variety of backgrounds and expectations that a community college curriculum is equivalent to the content of a university computing curriculum.

B. Curriculum

B.1. Increasing amounts of material that must be covered: As Moore's law reminds us, the complexity of the hardware doubles about every 18 months. This has been extrapolated by computer science teachers to be interpreted to mean that the complexity of what must be taught doubles in a similar manner. While this is a bit of an exaggeration, there is some truth to this. We need to look at more established liberal arts programs and see how they have managed to incorporate centuries of ideas and material. While their content is not changing as rapidly, it has existed for a longer time and much filtering has had to occur. In History and Philosophy, the goal is to identify the questions at each period. In computer science we need to identify the core material, realize that it is not really static, and put it into a formal curriculum. Emerging technologies can be handled in project classes and special topic classes until such time as they are viewed as core material.

If the approach to teaching is that we teach students some very basic, fundamental realities and then teach them to learn new technologies, (1) they will be prepared for a lifetime in a rapidly changing discipline and (2) the size of the core will be manageable and (3) the student will be ready to learn in a work environment.

B.2. Curriculum evolves as material moves down from the graduate level and up from the undergraduate level: In computer science, material tends to move rapidly from being research topics to being core material. Material also goes the other way. Compilers, for instance, were considered to be a necessary core topic in the 80's and 90's. Now they have been relegated to the ranks of graduate course work. While networking and distributed computing were originally part of graduate research, they are now considered to be essential to the undergraduate program. The role of the computing education professional then becomes one of discerning what material needs to be taught at what level and under what conditions material needs to be codified and moved to a different level.

B.3. Closing the Gap: The computing curricula taught in our academic programs tends to concentrate on the traditional CS knowledge base. While this education generally provides excellent background for the students to enter the profession, they are at a disadvantage in applying their knowledge to real applications. Often they are called upon to solve problems in business, medicine, science and technology applications for which they lack the depth of knowledge. It may also be necessary for them to function in multinational companies. While it is impossible to impart education in all application areas, some practical solutions can be pursued, such as preparing them for:

- Life-long learning
- Project management principles
- Multidisciplinary projects
- Communication skills
- Ethical and responsible behavior
- Team work
- Project-based courses (Computer Science in XXXX)
- Working in a global society

Some of this preparation can be addressed easily in seminars, while others can be accomplished by projects taught by teams of faculty and industrial partners working together. Industry colleagues can assist in this endeavor tremendously and will find it an excellent investment.

B.4. Exciting applications of Computing: At the outset, this may appear as a formidable challenge. Some of these applications, such as games, visualization, multimedia, immersion, real-time issues, ubiquitous computing, RFID technology, sensor systems, and service-oriented computing, are hard to introduce into an undergraduate curriculum. However, some of the faculty in CS and other departments may be working in these research areas. Therefore, we should take advantage of this rather than see it as a problem. Courses could be broken up into smaller modules. Faculty would then be able to teach their areas in the small. These class experiences could then be followed by a research internship in which students would be allowed to go into the faculty's laboratory and participate in the work. We can also depend on industrial internships as well as Research Experiences for Undergraduates (REU) as a mechanism to foster participation of students in such applications. Many industrial organizations would be willing to provide the hardware and software needed for such projects free of cost (or at a modest cost).

B.5. Conveying the excitement of new computing developments: Computing research is progressing rapidly, with streams of new applications and developments emerging from both universities and industry. Recent developments, such as portable video, music, cell phones, and game players, have captured the imagination of students and become required possessions. Computing departments are being challenged in how to convey the excitement associated with the new developments and the sense of accomplishment in constructing one of these applications, while teaching the more mundane fundamentals of the discipline. The exciting capabilities readily available and the fascinating and important problems are being obscured by the low-level aspects of curricula that have not changed in several years. Moreover, there are many critical application areas that require computing, for example healthcare, education, entertainment, transportation scheduling, security versus privacy issues, the environment, and defense. These areas are typically not addressed well within most curricula.

B.6. Using research to stimulate excitement: The new developments in computing represent an opportunity as well as a challenge. They can provide a motivation for students to learn the fundamentals. The needs of the new research areas and of industry can help to focus and direct the learning of more difficult topics. But, researchers must continually work to translate the advanced concepts of their projects into assignments that their undergraduates can tackle, thus engendering a feeling of contributing to current problems. This translation needs to bring the research concepts to the lowest levels of the curriculum, rather than having students wait until the end of their program before they get to do the new and inspiring things. Educators need to take advantage of the students' interests and strengths to motivate and engage them throughout their academic experience.

C. Graduates


One of the challenges in preparing undergraduates for computing careers is identifying what we want our graduating students to be able to do. The problem is that our students are going in different directions upon graduation. Most will be seeking immediate employment, while some will be going on to graduate school.

For those going on to graduate school, our task is not so daunting. A solid grounding in the fundamentals of computing, together with the skills to engage in advanced learning and research will serve these students.

Those seeking immediate employment present the biggest challenge, as it is envisioned they will find a wide range of different jobs. For these students the curriculum will need to incorporate a multitude of disciplines. We will no longer be able to teach the computing sciences independent of developments in areas such as health care, education, business, and criminal justice. The challenge will be to provide an environment wherein students can integrate the fundamentals of computing in the solution of application area problems. If we can succeed in this task, we may have taken a step toward addressing the problem of attracting students to the computing sciences, as students today are not interested in technology for its own sake. They are interested in what they can do with the technology.

D. Stakeholders


Designing strategies to catalyze the transformation of university computing education will require the active participation of a number of different stakeholders. The faculty must continue to grapple with the definition of a curriculum that will meet the needs of our students and their future employers. Today's students can be an invaluable source of information on how best to present the curriculum, in order to capture and hold the attention of students. Our alumni can provide information on the demands of industry, telling us how well their education prepared them for their jobs, which courses were most useful, and what kinds of courses they wish they might have had. Government, both state and federal, as well as other granting organizations, will need to provide funding to support our efforts to transform the curriculum and the educational environment. Funding will be needed to underwrite efforts to develop new models for teaching and curriculum and to create an infrastructure that will support new and emerging technologies.



Group 1 Presentation: PrePeparing Undergraduates for Computing Careers: *What are the Challenges?*

- n Incoming Students:
 - n Diversity of backgrounds and goals among incoming students, both K-12 and nontraditional
- n Curriculum:
 - n Increasing amounts of material that 'must' be covered
 - n Curriculum evolves as material moves down from the graduate level and up from the undergraduate level
 - n There is a huge gap between what is taught in computing curricula and what is practiced by computing professionals
 - n The exciting things underway in computing -- games, visualization, multimedia, immersion, real-time, ubiquity, RFID tags, and service-oriented computing -- are not reflected in most curricula
 - n How to convey the excitement associated with the new research areas and applications of computing
 - n Research challenges can help bridge the gap in interest and engender excitement
- n Graduates:
 - n Prepare students for graduate school or industry, and in specialized areas (business, healthcare, engineering, science, and social science)
 - n Prepare students for lifelong learning in computing

2/3/2006 4:58:21 PM



Who Might Participate

- n Government:
 - n Fund departmental level experiments in new models for teaching and curricula
 - n Fund infrastructure to support new technologies: robotics, sensors, networks, architectures, diverse platforms, etc.
- n Computing faculty: define the curricula
- n Alumni:
 - n Bridge/mediate between computing departments and industry demands
 - n Donate
- n Students: advise on presentation of curricula
- n Industry: articulate needs

2/3/2006 4:58:21 PM

II. Group 2 Report

Question 2 - Transforming the educational experience: *What might the community do to address the challenges?*

A. Assumed/Perceived Challenges:

- National decline in CS enrollments, lack of interest in computing among middle school and high school students, computing is perceived as synonymous to programming, boring, unattractive, geeky, lacking excitement
- We are not attracting enough of the brightest and the most intelligent and curious students
- Lack of public understanding what really the “computing” discipline is. Does the discipline encompass everything that needs computing to solve its problems more efficiently and cost effectively or the discipline involves the core theory and practice that enhances computing capabilities and others are viewed as “applications of computing”? Inclusive vs. Exclusive definition?
- Women participation has seen precipitous drop while that’s not true in other science disciplines or in engineering disciplines or in Mathematics
- Decline of US role in global competitiveness

B. Societal needs/Future Trends:

- Numerous applications areas offering significant long term challenges requiring traditional computing knowledge and knowledge in application domains to develop the infrastructure (equipment, software, information) to addressing the challenges. As these intersections broaden, more training in the intersections is required in order to function more effectively therein; problems unique to these intersections will emerge, requiring the understanding and study of new underlying principles of computing in these domains.
- Current and future students who seek careers in computing will continue to need a core knowledge of the fundamentals of hardware and software system design, but such core knowledge is not enough; deep knowledge of the domain of application of computing is now a co-requisite for effective problem-solving and discovery

C. Inhibitors:

- In high schools and for freshmen we teach programming in a language (syntax and all), not computer science let alone computing; we do not emphasize problem solving
- Current department structure in universities do not allow easy knowledge sharing and/or problem based learning
- We do not provide enough domain specific knowledge to our students to equip them to solve real life problems, do not provide enough relevance of computing to real life problems

D. What Needs be Done:

- Assess the graduates in terms of outcomes; they should have integrative experience in stead of doing things in a piecemeal manner, provide more application specific knowledge, team experience

- Provide more communication (verbal and written) skills, logical thinking skills, synthesis skills, reuse skills, deemphasize re-inventing wheels
- Reduce the core requirements to a minimal set, provide more flexibility to choose from a wide range of topics to complete degree requirements
- Use real life problem sets that are motivating, emphasize innovation
- Recognize that not all computing professionals need math or science to the same extent
- Encourage creativity, brainstorming, learning from mistakes, learning to learn by themselves, teach how to be adaptable
- Provide more design experience from different application domains
- Change granularity of courses to smaller modules, design prerequisite structure in terms of modules
- Provide training to think logically and precisely under pressure

E. Other Factors to Consider:

- 4 yr colleges vs. 2 yr. colleges – transfer issues, lifelong learning capability of students
- Recognize that students come from diverse social and economic backgrounds – some will need a more prescriptive approach than others
- Faculty motivation to retrain and/or to adapt – research vs. teaching faculty, faculty workload issues
- More adjunct faculty from practitioners
- Effect of changes in K-12 education and preparedness of students
- Influx of international students or lack thereof

Group 2 Presentation: Transforming the educational experience: *what might the community do to address the challenges?*

Group 2

Susan Bridges,
Adel Elmaghraby,
Richard Furuta,
Greg Peterson,
Pradip Srimani
and Oscar Garcia



Assumed/Perceived Challenges:

- Decline in CS enrollments, lack of interest, CS is programming, boring, unattractive, geeky, lacks excitement
- Not attracting the brightest and most intelligent
- What really the “computing” discipline is?
 - Does the discipline encompass all computing
 - Or the discipline involves the core theory
 - What is “applications of computing”?
 - Inclusive vs. Exclusive definition?
- Women participation is down!
- Decline of US role in global competitiveness

Societal needs/Future Trends:

- Long term challenges requires traditional computing knowledge and knowledge in application domains.
- Infrastructure (equipment, software, information) is needed to address new challenges.
- As the intersections broaden, more training in the other disciplines is needed.
- Students seeking careers in computing will continue to need core knowledge of hardware and software system design.
- Core knowledge is not enough; deep knowledge of the domain of application of computing is now a co-requisite for effective problem-solving and discovery.

Inhibitors:

- High schools and freshmen focus too much on programming languages details and not CS and problem solving
- Current department structure in universities do not allow easy knowledge sharing and/or problem based learning
- We do not provide enough domain specific knowledge to our students to equip them to solve real life problems, do not provide enough relevance of computing to real life problems

What needs be done:

- Outcomes-based assessment;
 - integrative experience in stead of doing things in a piecemeal manner,
 - more application specific knowledge,
 - team experience
- Include a sequence of multiple courses so that students can acquire and apply skills such as:
 - communication (verbal and written) skills (both in formal and informal settings),
 - logical thinking skills,
 - synthesis skills,
 - reuse skills, deemphasize re-inventing wheels.
- Reduce the core requirements to a minimal set, provide more flexibility
- Use real life problem sets that are motivating, emphasize innovation

What needs be done: (contd.)

- Recognize that not all computing professionals need math or science to the same extent
- Encourage creativity, brainstorming, learning from mistakes, learning to learn by themselves, teach how to be adaptable
- Provide more design experience from different application domains
- Change granularity of courses to smaller modules, design prerequisite structure in terms of modules
- Provide training to think logically and precisely under pressure
- Provide courses that provide multidisciplinary work and projects
- Provide multiple entry points in the programs to increase the diversity of incoming students.

Other Factors to consider:

- 4 yr colleges vs. 2 yr. colleges – transfer issues, lifelong learning capability of students
- Recognize that students come from diverse social and economic backgrounds – some will need a more prescriptive approach than others
- Faculty motivation to retrain and/or to adapt – research vs. teaching faculty, faculty workload issues
- More adjunct faculty from practitioners
- Effect of changes in K-12 education and preparedness of students
- Influx of international students or lack thereof
- Retraining and lifelong education of graduates
- Technology support for preparing students to enter computing curricula
- Distance learning should be considered as a tool and technology should be improved to provide distance learning especially for retraining and lifelong learning.

III. Group 3 Report

Question 3 - Models for transforming computing education: *What might an ideal undergraduate model for computing education look like in five years?*

A. Introduction

While it may be impossible to “predict” the precise role of computing in society in ten or even five years, our objective is to develop a model that itself is adaptable. In the proposed model, we define first a desired profile for students graduating with a computing baccalaureate degree. We believe that these proposed desired qualities will remain relevant for some time, although we do not want to rule out adjustment and even revolutionary changes to this initial proposed list. The second component of the model is to attempt to match the characteristics (both strengths and weaknesses) of students entering the program, with the desired qualities of graduating students. This exercise provides some clarity related to where more, or less, emphasis may be required in achieving the desired outcome. This process informs the delivery mechanisms that are used to implement the learning process. In particular, we encourage the use of techniques that resonate with the current generation of incoming students. Finally, any program, we believe, should have a core upon which the entire program relies. We have proposed a list of topics for a core. In summary, a key innovation of our proposed model is to leverage the characteristics of incoming students instead of force fitting mechanisms on current students that they perceive as irrelevant, while still meeting program outcomes.

B. Baccalaureate profile in computing

This document presents recommendations as a suggested model for transforming computing education at the baccalaureate level. We believe that any graduate from a baccalaureate program in computing must display the following attributes:

- (a) An ability to apply knowledge of computing, mathematics, the humanities, and social sciences appropriate to the discipline;
- (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
- (c) An ability to design, evolve, implement, and evaluate a computer-based system, process, component, or program to meet desired needs;
- (d) An ability to function effectively on teams to accomplish a common goal, display leadership, and mentor;
- (e) An understanding of professional, ethical and social responsibilities including the ability for self-evaluation;
- (f) An ability to communicate effectively with a range of audiences;
- (g) An ability to analyze the impact of computing on individuals, organizations and society, including ethical, legal, security and global policy issues;
- (h) Recognition of the need for, and an ability to engage in, continuing professional development and develop intellectual maturity;
- (i) An ability to use current techniques, skills, and tools necessary for computing practice;
- (j) An ability to create new products, services, processes, and jobs and for innovation and entrepreneurship;
- (k) General problem solving, critical/logical thinking
- (l) Personal insight

C. Matching the Abilities of the Net Generation with the Profile

The attributes in second 2 have been formulated with longevity in mind; however, the way in which programs are designed to allow students to achieve the objectives has to depend critically on the attributes of the incoming students.

We have identified certain characteristics of net generation that can be leveraged when they reach campus. These include team work, social interests, general technology skills, excellent communication skills albeit limited to peer to peer, and the use of technology to solve problems albeit in an ad hoc manner.

There are certain limitations that have to be overcome in order for students to successfully achieve programs outcomes. Examples include weak math skills and formal problem solving methodology.

D. Architectural Framework for Computing Curricula

Any curriculum model in computing must cover following areas

D.1 Computing Core

It is important that any program cover the core of computing, although different computing programs may vary as to the level of detail to which the core concepts are covered. We offer the following as suggested preliminary list of core topics

System Theory
 Language Theory
 Algorithm Design
 Hardware
 Information
 Representation
 Abstraction/Modeling
 Discrete Mathematics
 Statistics

We recommend that further inputs to any effort to define the core of computing be sought from the Ontology project: <http://what.csc.villanova.edu/twiki/bin/view/Main/OntologyProject>

D.2 Components

Every computing program must cover a set of components, that is, artifacts, resources, such as software, information, hardware, communication, people and organizations, and their interactions, as well concerns such as security and quality that transcend individual components. Again, the type of components covered and the depth to which they are covered will vary from computing discipline to computing discipline.

D.3 Methods

Any program must also cover appropriate methods provide depth of know-how and currency in the discipline.

D.4 System Level Integrative Experience within Selected Application Domain

Finally, the program must allow students the opportunity to integrate the knowledge that they have gathered, and include work on systems development, evaluation and integration in application domains.

E. Delivery Mechanism

We have tentatively identified two delivery mechanisms for achieving the program outcomes. We believe that both build on the strengths of the net generation, while addressing their weaknesses.

E.1 Continuing Interesting and realistic Project-Based Experiences

Project sequences begin in the initial course to introduce core topics. Later courses continue to utilize this project while reinforcing and expanding core topics. Appropriately planned projects will allow “situated learning” in which students are exposed to core concepts as and when needed. Students should also be allowed to repeatedly complete and improve projects under the guidance of a mentor, as this will enable students to develop reflective and self-assessment skills. Ideally, projects involve students from different levels in the program and from different disciplines.

The following barriers must be overcome to implement this learning experience:

Formulating performance expectations - Performance expectations will provide artifacts that can be used in the assessment process

Scheduling – The synchronization of this learning experience with more traditional course may prove difficult

Faculty load – Faculty effort is likely to increase significantly and it may, therefore, prove difficult to insure faculty buy in.

E.2 K-12 Outreach

These students will visit K-12 schools to demonstrate and explain their projects. This allows these students to utilize existing skills that they bring to the program while at the same time building and expanding other necessary skills. In particular, appropriate presentation skills including the creation of websites will be addressed. It provides an indication to prospective students of the excitement of computing and the expectations of students enrolled in computing programs. It therefore is likely to prove a highly successful recruitment tool.

F. Conclusion

This report has presented recommendations for transforming computing education at the baccalaureate level.

John Antonio

Mike Doran

Vladan Jovanovic

Han Reichgelt

Domingo Rodríguez

Ramón Vásquez

Group 3 Presentation

Summary Models for Transforming Computing Education

Han Reichgelt
Group Leader

Dallas, Texas, October 27-28, 2005

Group #3

Modified ABET CAC Outcomes

- u (a) An ability to apply knowledge of computing and mathematics appropriate to the discipline;
 - w Discrete Mathematics
 - w Statistics
- u (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution;
- u (c) An ability to design, evolve, implement, and evaluate a computer-based system, process, component, or program to meet desired needs;

Modified ABET CAC Outcomes (2)

- u (d) An ability to function effectively on teams to accomplish leadership, mentoring, and a common goal;
- u (e) An understanding of self-evaluation, professional, ethical and social responsibilities;
- u (f) An ability to communicate effectively with a range of audiences;
- u (g) An ability to analyze the impact of computing on individuals, organizations and society, including ethical, legal, security and global policy issues;

Modified ABET CAC Outcomes (3)

- u (h) Recognition of the need for intellectual maturity, and an ability to engage in, continuing professional development;
- u (i) An ability to use leadership, mentoring, current techniques, skills, and tools necessary for computing practice;
- u (j) An ability to create new products, services, processes, and jobs and for innovation and entrepreneurship;
- u (k) General problem solving, critical/logical thinking
- u (l) Personal insight

CORE

- u System Theory
- u Language Theory
- u Algorithm Design
- u Information Representation
- u Abstraction/Modeling
- u Discrete Mathematics
- u Statistics

Group 4 Report

Question 4 - Inhibitors and strategies: *Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA?*

A. Inhibitors with accompanying strategies

1. Lack of vision for computing disciplines for the future
 - a. Task force made up of visionary faculty from the five computing disciplines, industry representatives, government representatives and accreditation representatives should be established to create over-arching vision statement for computing of the future (10 years and beyond)
 - b. Ontology effort, Computing Curriculum 2005 and model curricula can be studied for current baselines and conversation starter
2. Lack of common core across current computing disciplines; fragmentation/silo effects have been created for separate areas – CS, IT, SwE, IS, CpE
 - a. The task force mentioned in #1 should establish a common core based on the future computing vision
3. Lack of funding for college/university implementation of Computing Curricula 2010-2015 recommendations
 - a. Establish funding from government and industry sources
 - b. Creation of the National Computing Foundation (NCF)
 - c. Connect with state/regional funding sources
4. Formal model curricula, Computing Curriculum 2005, IT Model Curriculum 2005, etc. should be on a continuous improvement cycle (5 years) – major and minor revision cycle
 - a. Professional organizations should assume this recommended cycle
 - b. Professional organizations should consider appropriate linkages with accreditation boards
5. Misconceptions about the fields and professions of computing, and unwarranted negativity of the profession keep students from considering and pursuing computing degrees and careers. Two of the areas that need to be addressed (especially for minorities and women) are the concepts of creativity in computing, and computing as a social activity that requires teamwork and collaborative activities in the work environment (such as assisting users).
 - a. Government agencies, industry partners and private foundations, such as the National Academy of Science should create and fund public service announcements that dispel the myths, rumors and negativities of computing careers. These announcements will be used to raise the awareness and excitement for careers in computing. We need positive role models including young people for these adds.
 - b. Creation of computer based instructional materials for elementary and middle schools to develop an awareness of the rewards of computing careers.
 - c. Local colleges/universities establish connections with middle and high schools to help educate teachers, counselors, students and parents concerning the computing

profession, career paths and the rewards of pursuing post-secondary computing education.

- d. Increase scholarships for students choosing computing majors by seeking external funding from business partners, advisory boards, professional organizations and private foundations.
6. Uneven quality of teacher preparation and enthusiasm for teaching math in K-12. Many elementary teachers convey their own insecurities, distaste and fears about math to the students. There is a lack of certified math teachers in middle and high school.
 - a. Colleges/universities need to address the math phobias for elementary teachers in their methods classes.
 - b. Schools need to increase the number of certified math teachers for middle and high school.
 - c. Colleges/universities need to offer courses for teacher education in logical thinking and problem solving.

James Cross
Sandeep Gupta
Phil Pfeiffer
Becky Rutherford
Cheryl Willis

Group 4 Presentation: Inhibitors and Strategies for Promoting Computing

James Cross
Cheryl Willis
Sandeep Gupta
Phil Pfeiffer
Becky Rutherford

28 October 2005

Inhibitor 1: A lack of vision for “computing” disciplines for the future

- **Establish task force to create an over-arching vision statement for the next-generation of computing**
 - membership: visionary faculty from the five computing disciplines, industry representatives, government representatives and accreditation representatives
 - time frame: 10 years and beyond
- **Study appropriate artifacts for current baselines and conversation starters.**
 - **Ontology Project** (what.csc.villanova.edu/twiki/bin/view/Main/OntologyProject)
 - **Computing Curriculum 2005**
 - **Other model curricula**

Inhibitor 2: Lack of common core across current computing disciplines

- fragmentation/silo effects for separate areas—CS, IT, SwE, IS, CpE
- Strategy
 - Use the task force mentioned in #1 to establish common core based on the vision.

Inhibitor 3: lack of funding for implementing Computing Curricula 2010-2015

- Establish funding from government and industry sources.
- Create a National Computing Foundation (NCF) to promote development of computing curricula.
- Establish connections with state/regional funding sources.

Inhibitor 4: failure of model curricula to track technological change

- Put formal model curricula on a 5-year continuous improvement cycle
 - Computing Curriculum 2005, IT Model Curriculum 2005, etc.
 - Major and minor revision cycles.
- Professional organizations should assume this recommended cycle.
- Professional organizations should consider appropriate linkages with accreditation boards.

Inhibitor 5: Misconceptions about the fields and professions of computing, and unwarranted negativity (1 of 2)

- Create and fund public service announcements that dispel the myths, rumors and negativities of computing careers.
 - Government agencies, industry partners and private foundations, such as the National Academy of Science
 - Must include vision of computing as a creative, social activity that requires teamwork and collaborative activities in the work environment (such as assisting users).
 - Goal: raise awareness and excitement for careers in computing.
 - Need positive role models including young people for these ads.

Inhibitor 5: Misconceptions about the fields and professions of computing, and unwarranted negativity

- Create computer based instructional materials for elementary and middle schools to develop an awareness of the rewards of computing careers.
- Establish connections with local colleges/universities and middle and high schools
 - Use to educate teachers, counselors, students and parents concerning the computing profession, career paths and rewards post-secondary computing education.
- Increase scholarships for students choosing computing majors
 - seeking external funding from business partners, advisory boards, professional organizations and private foundations.

Inhibitor 6: Uneven quality of teacher preparation, enthusiasm for math in K-12

- Many teachers convey insecurities, distaste and fears about math to students.
- Lack of certified math teachers in middle and high school.
- Strategies:
 - Colleges/universities need to address math phobias for elementary teachers in methods classes.
 - Schools need to increase the number of certified math teachers for middle and high school.
 - Colleges/universities need to offer courses for teacher ed. in logical thinking and problem solving.

Appendix C: Individual Reports

John Antonio – White Paper for ICER Workshop

1. *Preparing undergraduates for computing careers:* What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)? Encouraging CS faculty to become more engaged in commercial applications and issues. Often, faculty focus their energies on specific research problems, perhaps because they have no real institutional incentive to become more cognizant of commercial issues or concerns. This problem is not unique to CS, however, and is present in all disciplines of engineering.
2. *Transforming the educational experience:* What might the community do to address the challenges you identified above? Provide incentives for faculty by broadening the traditional definition of creative activity beyond the “single-PI, curiosity driven” research mode in which most faculty find ourselves. We have made progress in this direction at the University of Oklahoma by increasing the value of technology development activities.
3. *Models for transforming computing education:* What might an ideal undergraduate model for computing education look like in five years? I think it is important to not lose the fundamentals of the curriculum; concepts that have stood the test of time. However, an ideal curriculum also needs to instill a sense of confidence in the graduate. One approach in this direction that we are considering is to create a “Hot Technologies Course” that students take near the end of their program. This course would overview the latest and greatest tools and technologies being used in industry, and provide soon-to-be graduates the confidence to realize they can learn new technologies quickly. More involvement in multi-disciplinary capstone projects is another idea we are currently implementing.
4. *Inhibitors and strategies:* Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA? My sense over the past ten years is that kids in K-12 have the wrong idea about computing. Actually, I think it might be a mistake to introduce programming in high schools, because it gives a negative perception to many students about what computer science is all about. I have often found that well-rounded high-school graduates (with good math and science skills) perform better in our program than the stereotypical “high school hacker” that enters our program convinced she knows it all, and that there is nothing left for us to teach her.
5. *Who might participate:* What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation? What is the role of government in this process? Professional societies? Universities and faculty? Others? As I mentioned in item 1, I think it will be better to immerse faculty into commercial environments rather than the other way around. Often, when commercial stakeholders come to campus with curriculum ideas, it often agitates faculty, and creates barriers. This may be because commercial stakeholders have a more focused view of curriculum needs in the area of current technology, rather than the longer and broader view of faculty. I think the professional societies can help by marketing to the public in general what our discipline is, with an emphasis on problem solving (not just coding).

Integrative Computing Education and Research (ICER): Preparing IT Graduates for 2010 and Beyond

Susan Bridges
Department of Computer Science and Education
Mississippi State University

The focus of this paper will be on that part of the computing spectrum that encompasses the traditional disciplines of Electrical Engineering, Computer Engineering, and Computer Science, and the newer but closely related discipline of Software Engineering. These are often thought of as the most technical of the computing disciplines and at our land grant institution, like many others, are housed in the College of Engineering. Although we were asked to put aside notions of curricula, university education programs operate within specific curricula and evolve slowly. It may be that some new “disciplines” or “multi-disciplinary fields” will be added to university offerings, but I believe that most growth in computing will come, if it comes, in the traditional disciplines. Graduates from these degree programs are highly recruited and are predicted to continue to be in high demand for the next 10 years. However, the press coverage surrounding the burst of the dot-com bubble and outsourcing of computing jobs has led many prospective students to believe that the job market in this area is weak. Our most immediate challenge for educating students in computing is to convince them to enter the field. In order to do this we must convince them of two things: 1) there will be jobs available when they graduate and 2) the jobs will be interesting and fulfilling. In my opinion, the first will be much easier than the second.

Enrollment in computer science has plummeted in recent years, particularly for women and minorities. From my conversations with women and minority students, many of whom have entered one of our programs and then changed majors, the biggest problem is the “computer geek” factor. Several students have told me that computer science “destroyed my creativity.” The perception is that those who enter a career in computing will spend their days sitting in a cubicle writing code, will have limited interaction with other people, and will have little influence on people’s lives. This perception is reinforced by the way we teach our classes and by the current population of students we attract. We must take bold steps to overcome these perceptions if we are to attract not only the geeks (and make no mistake, we will continue to need the geeks) but also those students who want to change the world and want to spend their days interacting with other people.

We can use public relations campaigns to change some of the perceptions listed above, but we must fundamentally change the way we educate our students if we are to keep them interested in computing as a career, if we are to nourish innovation in the field, and if we are to adequately prepare them for computing careers. Although we have said for years that computer science is more than programming, most introductory courses in computer science are programming courses. The introductory courses that we teach have changed little in 25 years except that students now use laptops instead of submitting card decks and C++ or Java instead of Fortran or Pascal. One of the first things we should do

is to throw away all introductory computing textbooks. The example programs and assignments in most textbooks have changed very little. Who cares about the Towers of Hanoi? Introductory computing texts and introductory computing classes completely ignore innovations in computing such as GUIs, the World Wide Web, and email that have dramatically changed our lives. The use of computers for communication is not mentioned in our introductory courses even though this is one of the most pervasive uses of computing technology. Although we, like many other departments, now introduce software engineering concepts early in the curriculum, we only look at those aspects of software engineering that are directly related to programming.

The following are some suggestions for transforming computing education:

- Institute a first course in computing that includes but does not focus on programming.
- Emphasize the multidisciplinary nature of a computing career early in the curriculum by asking students to design computational solutions for problems in other disciplines.
- Use pair programming early and often in the curriculum.
- Team with faculty from other disciplines to teach “linked” courses. Examples might include biology and computing, physics and computing, accounting and computing.
- Require writing and oral presentations in all courses.
- Develop “fast track” sets of courses for people to transition from other disciplines into computer science.
- Integrate concepts from faculty research into undergraduate courses as soon as possible.

There are a number of factors that make it difficult to make substantial changes in undergraduate curricula. The demands on faculty time for research and publication at research universities make it difficult for faculty members to dedicate substantial amounts of time to innovation in undergraduate education. Even incremental change in curricula is can be difficult and dramatic change is much more challenging. Almost all computer science faculty members have painful memories of battles over the relatively unimportant topic of changing programming languages for introductory courses. It is more difficult to demonstrate that innovative curricula meet accreditation criteria.

The shortcomings listed above also define the stakeholders. Faculty and administrators must buy into the need for change and support efforts to implement change. This includes defining career paths for tenure track faculty whose major mission is improving education. Accrediting agencies must encourage and support innovation. Companies who are hiring computer graduates must be willing to hire students from non-traditional programs.

Integrative Computing Education and Research (ICER)
Preparing IT Graduates for 2010 and Beyond

Sheila E. Castañeda
Clarke College
Dubuque, Iowa 52001
cast@clarke.edu

The computing discipline has been faced with many challenges in its young existence. However, the kinds of problems facing us now are being compounded by societal and economic issues as never before.

1. Preparing undergraduates for computing careers: The biggest challenges facing educators are recruiting enough students into the computing field and keeping them interested and engaged once we have them enrolled. This is a daunting task when high school students, and their parents, hear about outsourcing, layoffs, long hours, and H-1B visas affecting career opportunities and prestige. Though students have ubiquitous access to technology and utilize it seemingly effortlessly, they do not understand how it works nor do they care – and they are not interested in trying to find out. Technology has become so mainstream to them that they look at it like other technological innovations that were marvels in their time, but have now become so common that we do not think about the technology that went into creating them – the automobile, television, phone, refrigerator, ... How many of us really understand how these technologies work – or want to take the time to find out – or even need to? We hire experts to fix them when they break down or just throw them out and buy the next better model. Students look at computer technology the same way, wanting to use these tools to solve problems or make their lives easier – but they do not want to study them to create new innovations.

2. Transforming the educational experience: To reach the millennial students, we must understand how students who were born after we began our careers learn best. These students work best with visual images (not text), crave interactivity, want inductive discovery rather than being told what they should know, have short attention spans, and move quickly from task to task without taking time to reflect. We will need to transform the ways we learned our disciplines to new curricula that are more experiential and collaborative, utilizing the strengths and characteristics of these young people to challenge them to make our society better by creating innovations in the computing field.

3. Models for transforming computing education: We can build on the characteristics of the millennial students to develop creative curricula that engage these students, such as utilizing graphics, games, robots, handheld devices, community-based projects. Because the millennial generation wants to responsibly solve societal problems, undergraduate education must include real-world applications for problem-solving. Expecting students to read large amounts of text is proving to be unrealistic – many students do not even buy the textbook, or only scan it for the outcomes and summaries provided. They use Google, Wikipedia and blogs to find out all the facts they think they need to know. Utilizing tools that build on students' visual preferences will be

important; but accomplishing this while balancing an ever-expanding knowledge base with computing basics in a finite curriculum structure is a huge challenge.

4. *Inhibitors and strategies:* The lack of highly qualified math and science teachers in grade school and high school is a national calamity. Without instilling an inherent interest in these fields at a young age, we will not be able to interest students in studying computing in college. This is especially alarming when we consider the number of women and minorities that are entering computing. We must make computing more friendly to these groups – not making aggressive game playing and high school hacking look like the prerequisites to studying computing in college and pursuing it as a life-long career.

5. *Who might participate:* It is going to take a concentrated effort by the entire nation to change the direction of computing. Industry will have to value a college degree and make employees feel respected. Grade school and high school teachers need to encourage more students, especially girls, to consider computer science in college. Education departments in colleges and universities need to encourage more future teachers to pursue certification in math and science. Computing departments need to engage students by focusing on students' strengths and talents within new curricula utilizing collaborative, experiential and socially responsible projects and close interaction with faculty members. Small colleges may be particularly well-suited to tackle this challenge due to small class sizes and personal contact with students. Colleges and universities need to encourage more students to go to graduate school to make sure we have enough future professors and innovators. NSF needs to support curricular changes in colleges and universities – even if the proposals are not innovative at the national level, they may be new at the local level. Model curricula need to be developed, with faculty development workshops taking place around the country to ensure adoption by the entire computer science community.

Preparing Computing Graduates for 2010 and Beyond White Paper (9/30/2005)

James Cross, Auburn University

1. **Preparing undergraduates for computing careers:** What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)?
 - a. **Attracting and Retaining Majors to Combat Declining Enrollments** – The U.S. Bureau of Labor Statistics has predicted *software engineers* to be one of the top ten fastest growing job occupations during the period 2002-2012 (see <http://www.bls.gov/news.release/ecopro.t04.htm>), yet the enrollments in computer science and similar programs are currently declining. The burst dot-com bubble and the extensive press regarding off-shore development of software have taken a toll on the available jobs for computing majors.
 - b. **Meeting Industry's Needs** – Perhaps a more important issue is whether our degree programs are meeting industry's needs. To this end, "off-shoring" of the middle of the software life-cycle, which is the very focus of many CS programs, will most likely continue to increase.
2. **Transforming the educational experience:** What might the community do to address the challenges you identified above?
 - a. **Attracting and Retaining Majors** – Efforts to attract students to the discipline may need to begin as early as middle school and continue through the first year of college. Many projects to address this issue are underway including efforts to broaden participation by minorities and women. However, the lack of retention of students that enter the program is somewhat more troubling. The first few "formative" courses can be critical to the success of students. At public institutions, experience indicates that as many as half of the students who begin in computing do not complete their degrees. While a portion of these are indeed in the wrong major, others just do not get "hooked" or simply cannot get past some hurdle in an early course. Twenty years ago, object-oriented programming and design, with its inheritance and polymorphism, were advanced topics. Now that these topics are commonly taught in the first course, care must be taken not to lose our prospective majors as a result of technical overload. Fortunately, pedagogically sound tools and environments that help reduce the complexity of these topics are becoming available. With a solid foundation in the first courses, students are more likely to be successful in the core software engineering courses such as software construction, design and modeling with UML, quality assurance, and process. A capstone senior design project, which is found in most curricula, should be one of the most important activities in the curriculum. This team-oriented activity is intended to emulate a professional experience as it draws together much of the student's previous coursework.
 - b. **Meeting Industry's Needs** – An industrial advisory board consisting of technical leaders and project managers can provide useful insight into current and future industry needs. In addition, carefully designed surveys for employers of current graduates can also be useful in determining the characteristics of an ideal graduate from the viewpoint of the company. As major industry players continue to off-shore detailed design and

implementation, we may have to focus our attention on upstream activities such as requirements analysis and software architecture, as well downstream user support activities such as system administration. Clearly, system admin jobs are not likely to be sent off-shore.

3. **Models for transforming computing education:** What might an ideal undergraduate model for computing education look like in five years?
 - a. **Multiple Degree Program Options for Students** – While other countries have had undergraduate software engineering degree programs in place for years, these are just beginning to be implemented and accredited in the U.S. IT degree programs are also being created at numerous institutions. However, the effect these new programs will have on enrollments is unclear. The *Computing Curricula 2005: The Overview Report* (http://www.acm.org/education/Draft_5-23-051.pdf) provides a description of CpE, SwE, CS, IS, and IT degree programs in general. Together these perhaps form the union of computing education. The report also attempts to describe the intersection of these degree programs. In the future, departments of CS may need to provide multiple degree programs which share a common core.
 - b. **Topics and Tools for the Future** – Many of the special topics courses in today's curricula will become part of the advanced core in the future. There will likely be increased reliance on managed code for the Java and .Net platforms. Platforms for software projects will include small devices such as PDAs and cellular phones, as well as embedded systems. As advanced architectures with multiple CPUs become commonplace, programming techniques for multithreading will need to receive more emphasis in upper level undergraduate courses. To make these topics less formidable for undergraduates, it will become increasingly important to make the use of appropriate software tools and environments an integral part of the courses. For example, special environments for cell phone software development and certification will be needed.
4. **Inhibitors and strategies:** Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA?
 - a. **Fragmentation in Computing Curricula** – While much good can be said about the new subdisciplines of computing, they do have a splintering effect with respect to traditional CS. Strategies directed at unifying the subdisciplines and mapping out ways for CS departments to reinvent themselves are clearly needed.
 - b. **Lack of Flexibility Among Faculty** – Tenured faculty may not be inclined to welcome new directions. However, if presented in the right unified format, they may become surprisingly more nimble with respect to new directions.
5. **Who might participate:** What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation? What is the role of government in this process? Professional societies? Universities and faculty? Others?

Clearly, government, professional societies, institutions, faculty, and industry must all be involved. The professional societies, which include representatives from both academia and industry, should provide the leadership while NSF provides funding for the effort.

Dr. Michael V. Doran, mdoran@usouthal.edu
Professor and Coordinator of Computer Science
University of South Alabama
School of Computer and Information Sciences
Mobile, Alabama 36688

1) Preparing undergraduates for computing careers:

With the recent downturn in enrollments we face many challenges. Perhaps the most critical occurs before our students reach our campus. K-12 now uses technology in many settings but seldom addresses the fundamental issues of computing which would encourage or prepare students to enter an undergraduate program of study. Students today know how to use tools and search the web or use PowerPoint, but are not exposed to how computing professionals create, manage and deploy these tools. It is almost as if a miracle occurs and the next version of software will appear. The career in computing that leads to these developments are seldom, if ever, discussed or considered. K-12 schools still teach the appropriate math and science to prepare students to enter the field; however, those students are often attracted to science and engineering since CIS has lost the luster it enjoyed in previous years. Now the numbers who consider CIS are down, and to compound the problem, they might not be the ones who have acquired the necessary math and science background to be successful. K-12 advisors know what is necessary to be a successful scientist or engineer and those students are guided without problem. Computing and technology have evolved and perhaps the K-12 understanding has not evolved at the same pace. It might be as simple as getting K-12 to again view CIS as a viable and growing field, and that it requires the same or similar K-12 preparation as engineering or most other sciences.

Once the students arrive on campus, we must retain and motivate them to be excited about a career in computing. The rigor of the discipline must be in place, but it must also be merged with the glitz and appeal of the use of modern technology. If students leave the discipline for academic reasons, as discussed above, that is often outside our control. It is unacceptable if a prepared and talented student leaves CIS for lack of interest in a static or stale curriculum and delivery of content.

2) Transforming the educational experience:

During the "Space Race" of the 1960's there was a national mission to advance science and technology. The field of computing enjoyed this national focus probably more than any other field of study. It is critical that again the nation renews and draws attention to the area of technology development. Some key endeavor should be identified which would unite education on a similar mission. Just as in the 1960's, when kids thought it was appealing to study and prepare for a career in science and technology, we must foster a new generation with the same ambition. A clear picture must be presented that without this renewed emphasis on computing, our national technological advantage will soon disappear (if it hasn't already). K-12 has the curriculum available in the

fundamental areas as well as the technology areas. Students must be motivated to undertake this career and have a renewed “national mission.”

3) Models for transforming computing education:

This is difficult to address. The body of knowledge grows each day. The core of necessary courses has long defined the established curriculum and can already fill more than four years of study. The need for breadth as well as depth continues to struggle for a balance. This is considered while still needing an ever diverse and broad general education curriculum of supporting areas in all disciplines. A practical approach might look to define an absolute core of central topics to computing. In the traditional curriculum this might include data/file structures, software engineering and language theory. Beyond that core, flexibility might be achieved by diverse focus tracks. Such focus tracks might include theory as found in many traditional curricula today. Other tracks could focus on more practical and/or appealing areas of computing such as: robotics, networks, game development, graphics, real-time systems, numerical methods, etc. This plan does not propose a vo-tech approach to CIS education, but rather the same rigor as present today carved into more focused areas of depth and practical courses. In order to assure some breadth, perhaps several focus areas might be required. Supporting and general education courses would likewise be crafted to coordinate focused learning in disciplines such as: science, business, education, psychology, economics, mathematics, etc.

4) Inhibitors and strategies:

Probably the main factor that drives most any successful thing in the country is the economy. Not being an expert in the field, it is hard to predict or understand the factors that influence it. As the recent hurricanes have proved, some economic factors are completely beyond anyone’s sphere of control. The main factor in our sphere of control that can impact CIS education is the ability to make the discipline attractive to K-12 students and have them understand how to be prepared to enter our discipline. With the turn around cited in much of the literature recently of the increasing demand for CIS professionals, the job market should help attract more students in the coming years.

5) Who might participate:

The usual stakeholders must participate and desire these changes to take place. Starting with the existing university community, the need for change must be identified and agreed to. Industry and other employment groups, which depend on qualified graduates from universities, must likewise recognize the need for change and work to bring it about. The stakeholders who control the financial direction of universities must participate. This includes local, state and national government and funding sources. Finally, the students are the main stakeholder and must be an integral part of the changing learning environment. The new environment must appeal to them and attract them to it. This transformed environment must meet their needs but not cater to unreasonable or unsound educational desires.

Alternative Perspective on Integrative Computing Education and Research

Adel S. Elmaghraby

adel@louisville.edu

Identifying the Challenges:

Understanding the challenges facing Computing education is essential. We all agree that a well-defined problem is much likely to be solved, and therefore highlighting some of these problems is needed.

- Too many conflicting goals in terms of managing our programs. These goals affect educational, research, and administrative efficiency and some examples include:
 - Various emphasis on teaching versus research – is there a perfect balance?
 - How can you justify professional service in times of limited resources?
 - Research quality versus research funding.
 - Accreditation focus on process as an alternative definition to “quality”.
- Employer needs are changing and employment period of an IT professional is getting too short. These changes require consideration of issues such as:
 - How much training versus fundamentals of the field.
 - Need for increased laboratory experience.
 - Growing needs for understanding legal, ethical and globalization issues.
- Better understanding of outsourcing and how it affects preparation of graduates.
- Basic funding of educational institutions is not necessarily at the desired baseline. This may have different reasons and levels at various institutions and may not hold for some, but for those who face it, these issues are of significance:
 - Increased pressure on generating soft money causes shifts in focus from long-term goals to immediately achievable opportunities.
 - Balance between full-time faculty, adjuncts, and graduate assistants in the classroom and a laboratory does not necessarily provide the best possible education.

Responding to the Challenges:

By following a structured approach, we can better respond to these challenges. The ICER workshop is certainly a step in this direction. Let me identify some steps:

- Identify goals, resources, and opportunities for our individual programs.
- Prioritize our goals and seek consensus among our stake holders.
- Seek alternative and innovative approaches that may leverage our resources and target our unique opportunities.
- Plan our strategies and “sell” our approach to those we need for support.

Considering Alternative Models:

By seeking alternative models, I would like to emphasize that these can be in relation to teaching, research, and service. Some examples of alternative models include:

- Partnerships with other institutions, departments, government agencies, industry, and alumni. These partnerships can be at a local, regional, national, and international dimension.
- Consider changes in the methods of delivering your education content such as:
 - Distance learning.
 - Web-assisted instruction.

- Lecture, lab, and discussion sections.
- Group learning activities and communities of learners.
- Leveraging independent and extra curricular learning activities.
- Studio Learning Approach and this, I will expand in more detail.

A Studio Model for Computer Education:

Observation of art and architecture models of studio-based education demonstrates that they provide students with immersive experiences that are much closer to reality as compared to traditional classroom and recitation approaches. In addition, the studio approach provides better communication and team skills – a much-valued outcome for the work environment.

Experiments have demonstrated usefulness of this approach to other disciplines in engineering [1] and science [2] and others have considered using it for post-graduate education [3]. So, is studio-based education relevant, desirable, or feasible for computing education?

My perspective is that is one of our best alternatives – several reasons support this argument:

- Current educational models are not well-suited for the job market.
- Increased challenges due to globalization require a more balanced and integrated understanding.
- Carving the curriculum in the traditional modular approach does not promote integration.
- Skills and theoretical fundamentals are better integrated and understood in a studio setting.
- Research experiences and industrial projects can be part of the studio curriculum and provide case studies and realistic learning challenges.

Concluding Remarks:

Achieving a studio-based approach is only one possibility towards an integrated computing education and research that can better adapt our graduates with the changing global environment. Partnership is essential to leverage resources and to improve the learning environment. Accreditation at various levels is providing, in some cases, an excessive overhead and has trivialized the measurements to focus on process and possibly miss on the more difficult to measure “quality” of experience and long term benefits to the graduates and the profession. Universities, communities and government agencies need to address education funding in a comprehensive way and include the students in the equations – innovations in funding students during undergraduate studies will allow them to spend adequate time in campus and have a comprehensive experience in a studio setting.

References:

1. The Influence of Technology on Engineering Education, by John R. (EDT) Bourne, A. (EDT) Brodersen, M. (EDT) Dawant. CRC Press, December 1, 1995 (page 98).
2. Improving Undergraduate Instruction in Science, Technology, Engineering, and Mathematics, Steering Committee on Criteria and Benchmarks for Increased Learning from Undergraduate Stem Instruction, Committee on Undergraduate Science Education, National Research Council - National Academies Press, January 1, 2004.
3. Design Education for Engineers, Lord St John of Fawsley - Royal Fine Art Commission, Publisher Thomas Telford, December 31, 1996.

Richard Furuta
Professor
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

Email: furuta@cs.tamu.edu

Computing education faces challenges related to external perception and to internal focus. Turning first to the issues of external perception, it is evident that in the broader community, computer science is viewed as synonymous with programming. Yet as the continuum of computing disciplines identified by the workshop's organizers illustrates, programming is only a single skill out of many that must be acquired by today's computing students. Beyond the recent enrollment drops, increasingly attributed to parental concerns about outsourcing of programming jobs, a longer-term concern is the effects of prospective students' perceptions as it restricts their interest in becoming involved with our field.

Computers, computing applications, and high-speed networks are commonplace in today's society. Nielsen//NetRatings, quoted in the *Chicago Tribune* of September 29, 2005, reports that 42% of all Americans now have access to broadband network connections at home. Nearly all K-12 schools are connected to the Internet. In my experience, for today's high school students, computers are a commodity and perhaps as a result, many view programming as a dull and tedious choice of career. Since computer programming is what is emphasized in high school classes (and indeed, in many college program's first year or two of courses), many students never have the opportunity to discover the true nature of computing careers.

Expressing the breadth of the scope of computing to prospective students is an important goal, but in addition we must take steps to reflect the breadth in our curriculum. The diagram provided by the workshop's organizers is informative in focusing thoughts about this, showing a continuum from "Electrical Engineering" on the left to "Users" on the right. Both ends are grounded in well-developed, and quite different, theories and traditions. The left draws from a mathematics-based theory and from engineering practice. The right draws from a theory derived from the social sciences and incorporates a tradition with strong influence from the arts. I think it is accurate to say that the common perception of Computer Science more strongly identifies it with the left side of the diagram rather than the right. Similarly, the students in most departments encounter the theory and practice from the left side of the diagram in introductory classes, moving to the right only in upper division elective classes.

While the perception that Computer Science is just about programming may have played a role in our recent enrollment declines, I think that the perception that Computer Science is primarily about the left hand side of the diagram has played a role in our longer-term decline in diversity. We are clearly appealing to only some of the students

with talents and interests of importance to employers in our field. Perhaps an important distinction might be to recognize that while it is critical that our students' education be well grounded in a discipline's theoretical framework, there most likely is not a *single* theoretical framework of relevance to computing.

There are many departments in the university with a primary focus on computing—some of the traditional ones include Computer Engineering, Computer Science, Information Science, and Management Information Systems. Perhaps the “right” way to address these issues will be to retain a status quo—for each department to focus on its traditional strengths—to give the student a palette of entry points to studying computing. An alternate approach within the context of Computer Science would be to modularize the classes offered so that a student could build a program that was centered in one of the several relevant areas of theory with supporting coverage of the others.

As a final point, I would like to note that beyond providing an equal footing to the range of areas affecting computing, computing programs must expand their scope to incorporate skills not emphasized traditionally. One such skill is writing proficiency. Increasingly, the focus of undergraduate engineering and computer science majors on purely technological topics is being called into question by employers, who point out the importance of effective written communication in the workplace. It goes without question that this skill is important for students who decide to pursue advanced degrees.

Aesthetic Computing in Computer Science Education
Paul Fishwick
CISE Department
University of Florida
fishwick@cise.ufl.edu

Introduction

Despite significant and long-term progress made by computer scientists over the past five decades, we are at a turning point in the discipline. There are several problems that have led to this juncture. The first problem is the most recent: students out of high school see less relevance in computer science as a major when they get to college. The causes for this are not clear, and theories range from outsourcing and the “dot com” economic bust to notions concerning how computing is viewed as generally less relevant or important to real life when contrasted against other areas such as the biological and medical sciences.

Part of the problem rests with the observation that computer science is fairly abstract, almost by definition. Abstraction, while central to mathematics and computer science, requires a balance with reification. Abstraction in the form of symbol mapping does not imply “abstracting away the senses.” The reification process has actually been occurring for many years in the form of new metaphors for computation. By adding concepts like classes, objects, inheritance, and agents into software engineering, we think differently—through analogy and metaphor—about how computing is defined. The user interface adds to the collection, with loosely defined metaphors for a functioning office (i.e., folders, copying, printing, desktop) dominating the scene for the past twenty years. However, the use of metaphors is not sufficient, unless such use is accompanied by reified objects (pictures of folders and trash cans). We might ask ourselves why we have not taken this metaphorical evolution back to programming language development, or perhaps even further to mathematics itself? To some extent, we have. There is a rich history of visual programming, and curiously enough it is beginning to finally surface in mainstream computing in the guise of model-driven architecture (MDA). Another area that seems to hold significant promise is “serious gaming” where game-based approaches are used to teach computer science. If we can weave gaming into the curriculum, this partially addresses the reification problem and hopefully motivates students to program and learn more about computer science.

Our approach has been to consider reification, gaming, and visual programming and then to ask ourselves “where we might go from here?” The elements of gaming, graphical user interfaces, and visualization involve a celebration of the senses: seeing, touching, hearing, and the unifying field that combines them is *art*. By applying *art as method* (i.e., aesthetics) to model and program representation, we can manifest

structures that can be anything from games and cinema to designs, spaces, and sculptures. This new emphasis is termed *aesthetic computing*, and we have been working on it for the past six years.

Aesthetic Computing

In 2001, we obtained a grant from the National Science Foundation, EIA-0119532, entitled "An Investigation into Aesthetic Computing within the Digital Arts and Sciences Curricula." We have been teaching the course each spring semester. Students learn a method of starting with a formal mathematical expression, computer program, or data structure. They then take this structure through a set of transformations to yield a final representation. The three types of representations are *2D*, *3D*, and *Physical*, with the physical representation being shown for one week in a public gallery toward the end of the semester. The method has also been employed for a year in a class in Computer Simulation at the undergraduate and graduate levels. In this class, the approach is to teach simulation skills using art and design as catalysts for creativity. Model *structure* and *behavior* are subject to the creative skills of the students.

In September 2005, we held a workshop for mathematics teachers who teach algebra, with the idea that aesthetic computing can be used to 1) create a structured way of organizing algebraic expressions (in concept maps), and 2) introduce qualitative benefits into the classroom including improved student motivation, interdisciplinary connections with art and English, and an emphasis on individualization.

Bibliography

Fishwick, P. (2002) "Aesthetic Computing: Crafting Personalized Software", *Leonardo*, MIT Press 35 (4), pp. 383-390.

Fishwick, P., Diehl, S., Prophet, J., and Lowgren, J. (2005) "Perspectives in Aesthetic Computing", *Leonardo*, MIT Press, 38 (2), pp. 133-141.

Fishwick, P. ,Ed (2006), *Aesthetic Computing*, MIT Press, due in early 2006.

Aesthetic Computing Course Site: <http://www.cise.ufl.edu/~fishwick/aescomputing>

Math Ed Web Site: <http://www.cise.ufl.edu/~fishwick/acworkshop>

Guiding Principles for Redesigning Computer Science Education for the 21st Century – A White Paper

Sandeep K. S. Gupta (Sandeep.gupta@asu.edu), Department of Computer Sc. & Engg., Fulton School of Engineering, Arizona State University, Tempe AZ 85287. <http://impact.asu.edu>.

1. “Computer Science Educators – We have a Problem!”

A recent May 2005 CRA article “Computing, We Have Problem” by Jim Foley, CRA Board Chair, identifies a current problem in Computer Science (CS) education – a trend towards lower enrollments in many undergraduate computing programs. In the same CRA issue, the article “Challenges for Computing Research”, by Peter Freeman and Lawrence H. Landweber from NSF, identifies an “increasing competition for international students” due to “upsurge in the economies of countries, such as India and China and the attendant professional opportunities for students who remain at home, increasing numbers have opted for local universities”. This is partly due to the recent trends of off-shoring numerous programming related jobs to these countries to tap their cheaper workforce. And while the potential CS enrollments are falling, the projected demand for CS graduates is on the rise due to expected increase in the use of computing in all areas such as healthcare and security. As indicated in Foley’s article, computing education has an “image problem” due to its focus on programming – an activity which is mainly associated with nerds and hackers. How did we get here and what is a way out? To quote Einstein “We cannot solve our problems with the same thinking we used when we created them.” In order to address the problems created by off-shoring of CS jobs and decrease in CS enrollment there is a need to rethink CS pedagogy.

Early problems in CS education were pointed by D. L. Parnas (DLP) specifically “... top industry researchers and implementers ... prefer to take engineers or mathematicians, even history majors, and teach them programming.”[LDP90]. However, such concerns were debunked partly due to availability of umpteen jobs for CS graduate first due to Y2K problem and then the arrival of Dotcom era. However, the problems so clearly stated in DLP’s article and the doubts about nature of computer science [see ACM Computing Surveys 27(1) March 1995] are now resurfacing. It is time to take these concerns seriously in light of current problems. A serious redesign of CS curriculum can serve as a “Rite of Passage” to a more mature and respected discipline. In the following **four guiding principles** for redesigning CS education are discussed:

1. Move away from “reductionistic” approach to “holistic” - a more systems oriented - approach.
2. Emphasize convergence and identification of “fundamental principles” and merging of traditional courses.
3. Revise curriculum to reflect Dave Patterson’s SPUR (Security&Privacy, Usability, Reliability) manifesto [DAP05].
4. Evolve CSE curriculum to incorporate fundamentals of important emerging areas such as biomedical informatics.

2. Holistic or Systems Oriented Educational Approach

According to E. W. Dijkstra (EWD), “Computer Science is no more about computers as astronomy is about telescope.” This profound statement about the nature of computer science has many implications on how it ought to be taught. Some of his ideas on CS pedagogy are stated in his article “The cruelty of really teaching computer science” which had sparked an early debate on teaching computing science [PD89]. This article points a fundamental problem in today’s reductionist approach to computer science education – if a student can understand all its parts – OS, compilers, PL, Database Systems then he or she can understand how a computing system functions as a whole. Incidences such as various security attacks on computer systems point towards the needs to pay more attention to the interaction of various components in a computer system. As rightly indicated by EWD, a computer scientist needs to think holistically – a more systems thinking approach – with emphasis on the whole system functionality and the interdependence of its various components. EDW’s ideas about how to teach computer science are debatable due to his believe that computer programming is a branch of mathematics and his exclusive emphasis on training CS students in formal logic. Nevertheless, it is a fact that computer systems are becoming increasingly complex. Multiple agents (both within a computer and distributed across a network and maybe within the network) interact in various ways – P2P, client server, etc. – with dynamic trust relationships. Overall behavior of a networked computer system cannot be simply understood by understanding its parts individually. This behooves us - educators - to ensure that the computer science curriculum and pedagogy evolves to train the future generations of computer science professionals in systems oriented (top-down) methods.

One possible way to improve the image of CS – as a field for training programmers - may be to take a “top down approach” with an “inverted curriculum” similar to B. Meyer’s (BM) idea for teaching software engineering (SE) [BM2001]. Observations made by BM in the context of SE education are very pertinent to overall CS education in general and systems education in particular. The right emphasis on principles, practices, applications, tools, and mathematics are essential for a well-balanced university education which aims to provide not just vocational training to the student but prepare them for ever evolving field like computer science.

3. Need for Convergence

Thanks to proliferation of computers in scientific research, the sciences are seeing convergence [GWF01]. However, CS itself seems to be resisting convergence of its traditionally core areas. An as example consider how systems software courses (OS, compiler, networking) are currently taught. In most CS programs, there are dedicated courses on OS, compilers, and networking. OS courses focus on topics such process and memory management, synchronization primitives, and access control. Compiler courses teach lexical analysis, parsing, and code generation and optimization. Many of the networking courses teach the seven layer architecture covering protocol details at each layer. However, a typical CSE graduate is neither going to be writing OS, compilers, nor networking protocols. Then what is the rationale for having these courses in an

undergraduate CS curriculum? Can we extract the essence of these traditional courses – e.g. concurrency, layered architecture, syntax and semantic analysis – and merge them into fewer, more integrative courses?

On the other hand there seems to be a lack of emphasis on the fundamentals. Books such as Keshav's book on networking [Kesh98] have identified common design techniques for computer systems – but this is a rarity as opposed to common practice. Normally, undergraduate networking books focus too much on protocol details and do not pay much emphasis on fundamental principle such as end-to-end principle for layering [SRC84, MBDC01]. Hints on computer systems designs have been only informally stated [BWL83] and there is a need for formalizing them and teaching them to undergraduate students. This will only happen if we take a more holistic approach to teaching computer science. In fact a new balance between reductionist and holistic approach should be forged – a kind of balance between Yin and Yang [ACS98].

4. SPURing CS Curriculum

We are seeing the proliferation of C&C (computer and communication) devices - thanks to the push for faster and cheaper C&C in the last century [DAP05]. However, the cost of operating these devices (in terms of end user involvement and worries) has dramatically increased. We hear daily of identity theft cases due to someone breaking into a credit card database. One has to deal with distributed storage in multiple devices. Lack of penetration of information technology in important area such as Healthcare is due to privacy and reliability concerns. If we have to solve these problems the CSE curriculum has to change to more seriously address Security, Privacy, Usability, and Reliability (Dependability). New software engineering approaches such as Trusted Components [BM03] which have great potential to revolutionize how dependable software is developed in the context of off-shoring and security concerns. Ubiquitous computing and bio and health related topics are common recurring themes in a recent "Grand Challenges in Computing Research" by T. Hoare and R. Milner (eds) [THRM04]. We need to train our undergraduates to ensure that they understand these challenges – since they are the one who will be involved in developing the solutions.

5. Need for Evolution – Bridging CS and Other Sciences & Engineering

Since its inception, the modern computers (as simulation tool) have revolutionized (revitalized) many fields by bringing together theoreticians and experimentalist and enabling virtual explorations at vast spatio-temporal scales. Computers have been used as a "[Silicon] Laboratory and a Metaphor for Understanding the Universe" [GWF01]. Current interest in bioinformatics is one such area where computers are helping in understanding the "human code" in the quest to address various maladies including cancer and aging that affect humans. As the US population ages there would be increasing shortage of health care professionals. A recent article in the Time Magazine (<http://www.time.com/time/magazine/article/0,9171,1074139,00.html>) on e-health, on-going National Academy of Engineering project on "engineering the delivery of Health Care" (<http://www.nae.edu/nae/naepcms.nsf/weblinks/MKEZ-5LWQVN?OpenDocument>), a recent medical data bill by Senator Clinton and Frist (<http://www.tennessean.com/apps/pbcs.dll/article?Date=20050617&Category=NEWS08&ArtNo=506170412&SectionCat=NEWS&Template=printart>), and workshops such High-Confidence Medical Devices Software and Systems (HCMDSS <http://www.cis.upenn.edu/hcmdss/index.php3>) all identify the urgent need for increasing the use of information technology in healthcare. Currently, SPUR technology which would form the backbone of healthcare delivery is virtually non-existent. Who would want a networked medical device which can be infected by computer viruses and fail at inopportune time? A fundamental question to ask is how the future generation of CS undergraduates can be educated so that they can help in development of SPUR technology in other fields. Imparting CS students knowledge of the fundamentals of other science and engineering fields would make them more appealing for employers developing computing based solutions for fields such as healthcare.

6. Conclusions

CS undergraduate curriculum needs to be redesigned to address the dramatic changes in the role of computers in the society, the "negative" image of computer science profession, and increasing importance of computers in emerging areas such as bioinformatics and e-health. This paper identified four principles for redesigning CS undergraduate curriculum – a way to reach a more balanced and refocused SPURized curriculum with stronger bridge to other engineering fields and sciences.

References

- EWD88 E. W. Dijkstra, The cruelty of really teaching computer science, 1988.
<http://www.cs.utexas.edu/users/EWD/ewd10xx/EWD1036.PDF>
- THRM04 T. Hoare and R. Milner (Eds), Grand Challenges in Computing Research, The British Computer Society, 2004.
- BM01 B. Meyer, Software Engineering in the Academy, IEEE Computer, pp 28-35, May 2001.
- BM03 B. Meyer, The Grand Challenge of Trusted Components, May 2003.
- GWF01 G. W. Flake, The Computational Beauty of Nature – Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptations, MIT Press, 2001.
- DAP05 D. A. Patterson, 20th Century vs. 21st Century C&C: The SPUR Manifesto, CACM 48(3), Mar. 2005.
- Kesh98 S. Keshav, An Engineering Approach to Computer Networking, Addison Wesley, 1998.
- SRC84 J. Slatzer, D. Reed, and D. D. Clark, End-to-End arguments in systems design, ACM ToCS, 2(4), 1984.
- MBDC01 M. Blumenthal and D. Clark, Rethinking the Design of the Internet: The End-to-End Arguments vs. the Brave New World, ACM Transactions on Internet Technology, Aug. 2001.
- BWL83 B. W. Lampson, Hints of Computer Systems Design, 9th ACM Symp. on OS Principles, Oct. 1983.

- ACS98 A. C. Sodan, Yin and Yang in Computer Science, CACM, April, 1998.
PD89 P. J. Denning, A Debate on Teaching Computing Science, CACM 32(12), Dec. 1989.
LDP90 D. L. Parnas, Education for Computing Professionals, IEEE Computer 23(1), 1990.
DAP04 D. A. Patterson, The Health of Research Conferences and the Dearth of Big Idea Papers, CACM, 47(12), 2004.

Computing for Everyone: Improving Global Competitiveness and Understanding of the World
A White Paper for ICER Workshop
Mark Guzdial, Georgia Institute of Technology

The goal of computing education for the next five to ten years is to ***establish computing as part of a liberal, general education***. Like mathematics or laboratory sciences, taking computing courses should be a presumption of an educated professional or academic. Everyone would take some computing, and most would take more.

There are three reasons for the necessity of this goal. The first is that that's where the future jobs are, in the mix of computing with other disciplines. As Thomas Friedman argues in his book *The World is Flat* (Friedman, 2005), forces of global competitiveness require future workers to become *Versatilists*. "The world 'versatilist' was coined by Gartner Inc., the technology consultants, to describe the trend in the information technology world away from specialization and toward employees who are more adaptable and versatile...Enterprises that focus on technical aptitude alone will fail to align workforce performance with business value." Even technological powerhouses like Microsoft is looking for versatilists, as Bill Gates said, "The nature of these jobs is not closing the door and coding. The great missing skill is somebody who's good at understanding engineering and bridges that to working with customers and marketing (Montalbano, 2005)."

The second reason is that a liberal education is about understanding one's world, and computing is a huge part of today's world. We ask students to take laboratory sciences (like biology, chemistry, and physics) in order to better understand their world and to learn the scientific method for learning more about their world. The virtual world is an enormous part of the daily lives of today's professionals. Understanding computing is at least as important to today's students as understanding photosynthesis.

The third and most significant reason is to meet the potential of computing and other fields. Alan Perlis first called for computing as part of a liberal education in 1961 (Greenberger, 1962). He argued that the ability to specify and execute process would offer a whole new method of exploring domains and learning. If you understand something well, you should be able to define its process well enough for a machine to execute it. If you can't, or the execution doesn't match the observed behavior, we have a new kind of feedback on our theories. He used examples from economics in his 1961 talk, but the entire field of computational science demonstrates his prescience. Computing has enormous potential in many fields where its use today is limited to whatever Microsoft Office can do.

Computers are workhorses for plowing mental fields, but to harness these beasts of burden, one has to know how to command them. Most professionals today are limited to using applications developed by technology specialists, who can't possibly understand all other professions as well as their practitioners. To follow the analogy, it's as if farmers and mill owners of the past were told, "Look, horses are only good for pulling wagons for merchants. I can't help you with your plows and mills, but I can sell you wagon." The potential for computing in our society will only be met when all professionals have the capacity for understanding and commanding computing workhorses. When educated people across our society realize what computing really can do for them, the demand for software development professionals will increase to scale and disseminate the good ideas of the versatilists.

Our current computing education cannot meet this goal. Our track record for educating students about computing is dismal. We can't attract and retain the students who claim that they

want to focus on computing, and it's much worse with non-majors. The percentage of women and minorities taking computing courses continues to drop, even with all the attention paid to it.

The largest change that must occur in our computing education in order to create computing for everyone is to recognize that the goal of computing education is not *only* to produce software development professionals. Creating software development professionals will be a fraction of the challenge of our education task if everyone on every campus studied computing. We overemphasize techniques and methods for large scale software development in our classes, which are not the most important benefits that we have to offer the rest of academia.

A computing for everyone should emphasize the laws, limits, uses, and wonders of computing. A few examples include:

- That we can define better or worse processes, and that processes can be *proven* correct (something that Perlis thought everyone should be taught).
- That there are processes that can't be successfully defined like a solution to the Halting Problem, or if defined for a computer to execute, may not finish in your lifetime.
- That the line between 'program' and 'data' is permeable, and that exploiting that permeable boundary is how many viruses attack.
- That information, once digitized, can be mapped and re-encoded into other media, forms, and representations.

Our proof of concept is the new courses at Georgia Tech's College of Computing. We have been creating *contextualized computing education* where we teach non-CS majors in classes that draw on relevant examples and uses in their field and that emphasize computing concepts and skills that go beyond just software development. We teach engineering students MATLAB with engineering-oriented problems, and we teach management, architecture, and liberal arts students computing for creating and manipulating media (Guzdial, 2003). We are enjoying dramatically higher retention rates in these contextualized courses than in more traditional computing courses, with women and minorities succeeding at the same rates as white men (Rich *et al.*, 2004). It's a portable innovation: the courses are being adopted at other institutions with similar improvements in retention (Tew *et al.*, 2005). But most importantly, follow-up studies have students telling us a year later that the courses have changed how they think about computing and use it in their daily lives (Guzdial & Forte, 2005). GT's new *BS in Computational Media* degree, a versatilist combination of computing and liberal arts, drew over 100 majors in the first year and is nearly one-quarter female. Our field needs similar innovations in many contexts to draw in everyone across the academy into computing.

References

- Friedman, T. L. (2005). *The world is flat: A brief history of the twenty-first century*: Farrar, Straus, and Giroux.
- Greenberger, M. (1962). *Computers and the world of the future*: MIT Press.
- Guzdial, M. (2003). *A media computation course for non-majors*. Paper presented at the Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE) 2003 Conference.
- Guzdial, M., & Forte, A. (2005). Design process for a non-majors computing course, *Proceedings of the 36th SIGCSE technical symposium on Computer science education*. St. Louis, Missouri, USA: ACM Press.
- Montalbano, E. (2005, July 18). Gates worried over decline in us computer scientists. *Computer World*.
- Rich, L., Perry, H., & Guzdial, M. (2004). *A CS1 course designed to address interests of women*. Paper presented at the Proceedings of the ACM SIGCSE Conference.
- Tew, A. E., Fowler, C., & Guzdial, M. (2005). Tracking an innovation in introductory cs education from a research university to a two-year college, *Proceedings of the 36th SIGCSE technical symposium on Computer science education*. St. Louis, Missouri, USA: ACM Press.

Mirsad Hadzikadic
University of North Carolina - Charlotte

1. *Preparing undergraduates for computing careers: What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)?*

Understanding the true needs of the industry/businesses is exceedingly hard (despite the usual claims that “softer skills like communication, team focus, and problem solving” is what is needed). They want graduates who understand both business and computing. However, they are not willing to define, in terms of skill set, what will be needed long-term. Part of the problem is that their business needs/goals change quickly. The short-term profits, not long-term planning, drive their decisions. It is hard to promote IT as a field under those circumstances (job security issues, etc.).

Under-representation of women and minorities in IT is troublesome. IT is still seen as a career for “nerds.” The key to the success here might be in accepting and promoting IT as an interdisciplinary field. IT is becoming an integral component of so many disciplines. Why not teach it that way?

Separating fads from true “sub-disciplines” is getting harder, primarily because of (a) the speed of changes in technology itself and (b) the degree of integration between technology and business processes.

2. *Transforming the educational experience: What might the community do to address the challenges you identified above?*

Form partnership with universities. Help us define the curriculum, with an eye toward long-term. Help with funding (especially in the case of state universities).

3. *Models for transforming computing education: What might an ideal undergraduate model for computing education look like in five years?*

I believe that the key words will be: flexibility; interdisciplinary, customized curricula; hands-on experience; highly integrated internships.

4. *Inhibitors and strategies: Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA?*

Clearly, the funding is an issue. However, equally important are: the quality of secondary education (math and science), emphasis on quality teaching, modifying the concept of tenure, and re-thinking the hierarchical structure (disciplinary silos) of universities.

5. *Who might participate: What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation? What is the role of government in this process? Professional societies? Universities and faculty? Others?*

I do not believe that it is up to any particular committee to sort things out. We simply need to let universities respond directly to the changing needs of the society without any restrictions imposed by the accreditation bodies. Several interesting ideas will emerge. The problem will be in insuring that the public is aware of the differences in the quality of offered solutions and offered education/degrees.

ICER Workshop White Paper

Michael N. Huhns

Department of Computer Science and Engineering

University of South Carolina

Huhns@sc.edu

There is a huge gap between what is taught in computing curricula and what is practiced by computing professionals. (This problem is not unique to computing education: in electrical engineering there is a similar gap between a traditional course in circuit design and the way that electrical/electronic components and systems are designed and constructed.) Very few practitioners write individual Java programs, as is taught in CS1, or write compilers, as in a senior course on compilers, or design digital circuits consisting of a few gates and flip-flops, as taught in a sophomore course on digital logic design.

There are many exciting things underway in computing: games, visualization, multimedia, immersion, real-time, ubiquity, RFID tags, and service-oriented computing. These are not reflected in most curricula.

There are many critical needs that require computing: healthcare, education, transportation (flight rescheduling), entertainment, security vs. privacy, the environment, and defense. These are not addressed by most curricula.

The Web provides massive amounts of information, but it is impoverished in its connection to sensors and effectors. This will soon change, and curricula must support this.

1. *Preparing undergraduates for computing careers:* What are the biggest challenges that you face in your role (i.e., as an educator, employer, administrator, leader, other)?

As an educator, my biggest challenge is conveying the excitement of new computing applications and the sense of accomplishment in constructing one of these applications, while trying to teach where semicolons belong in Java. The exciting capabilities readily available and the fascinating and important problems are being obscured by the low-level and mundane aspects of curricula that have not changed in several years.

We are on the brink of major changes in computing, brought about by two hardware advances—unlimited and ubiquitous bandwidth and multicore processors—and one software advance—service-oriented computing. These advances will cause profound changes not only in how computer applications are developed and executed, but also in the very nature of the applications themselves. Computing curricula should be leading, not lagging the changes. Unfortunately, the number of things that seemingly *ought* to be taught in an undergraduate computing curriculum is increasing, but the time available for the curriculum is fixed.

2. *Transforming the educational experience:* What might the community do to address the challenges you identified above?

It would be helpful if there were community-wide consensus about the necessary fundamentals for an undergraduate computing curriculum that reflected the new capabilities in hardware and software and that narrowed the gap between the curriculum and practice.

3. *Models for transforming computing education:* What might an ideal undergraduate model for computing education look like in five years?

- Software development will be taught more from a methodologies standpoint. This will enable the consideration of testing, maintenance, and security from first principles.
- Both hardware and software will emphasize the role of development teams and a consideration of the social aspects of both implementation and use.
- Programming will distinguish between client-side and server-side systems, and will emphasize the role of services as an enabler for “programming-in-the-large.”
- Theory will remain largely unchanged.
- Operating system and compiler courses will become graduate specialties.

4. *Inhibitors and strategies:* Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education? Can you identify strategies that may enable the transformation of undergraduate computing education in the USA?

The major inhibitors are (1) the lack of course materials to support the new computing capabilities, (2) the cost for departments to acquire the multimedia, sensing, and robotic/effecting systems, and (3) simply the inertia within the educational system to revamp the curricula.

A strategy is to first develop a model curriculum that reflects the new computing attributes and capabilities. Once it becomes widely adopted, there will be an incentive for new materials to be produced. But, state and federal agencies must be convinced to provide the requisite funding.

5. *Who might participate:* What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation? What is the role of government in this process? Professional societies? Universities and faculty? Others?

All of the above have a role to play, especially because it is in their vital interests to do so. However, the major impetus and effort must come from university educators.

Vladan Jovanovic, College of IT, Georgia Southern University

Developing Student Potential: The Framework for Education in IT

This ICER white paper identifies direction for IT education and characterizes an Architectural Framework for a Research Oriented Model of Evolving Computing (IT) Curricula. The world is changing by automation of routine work by IT and this can only intensify. The only viable response for Universities is fostering challenging environment for intellectual pursuits conducive to the full development of students.

By 2010 evolving IT disciplines will be sufficiently established, increasingly experienced in working together, and probably shifting more efforts towards direct student involvement in a shared multidisciplinary research. One additional assumption is that improvement and sustainability oriented mindset will replace the ‘next killer application’ or the ‘silver bullet’ mentality.

Challenge: Focus IT educators/leaders of educational establishments decisively on increasing their involvement in relevant research with students, in order for students to become more reflective and experienced practitioners in use of scientific methods and industry best practices. Direct consequence of such focus on development of student potential will be more mature, able and competitive students. Integration and adaptation abilities in a changing world require mature students with cultivated insight. IT education must address nurturing of these abilities and the premier way to do so is exposure to variety of viewpoints in a challenging environment.

Responding Strategy: Framework for architecting an integrated research oriented curriculum in computing. Proposed framework covers: Outcomes as requirements, Learning Areas as components, Policies as enabling mechanisms, and identifies the need for feedback from students Pre and Post University. Setting the stage for the new faculty-student collaboration includes recognizing:

- a. Grand challenge areas in improving human conditions to be addressed like Life Information Processes and Semantic Web
- b. Important Goals regarding students development, specifically: Character, Maturity, and Capabilities for research of practice and of systems design
- c. Principles include: Relevancy, Involvement, Flexibility, and Consolidation. For example (consolidated) curriculum should explore significant overlap of fundamentals for all IT disciplines and effectively incorporate missing elements of systems and information theory, focus on knowledge representation, integrate theory with solved problems and deliver most fundamentals early.

Outline of the proposed Framework for Architecting Computing Curricula follows:

1 Outcomes as explicit Maturity and Capability Requirements

- i. Understanding of systems and of technology
- ii. Personal Insight
- iii. Professional Insight

- iv. Careful and logical thinking
- v. Fast acquisition of knowledge and skills
- vi. Communication and persuasiveness
- vii. Experience in solving application domain problems
- viii. Organizational skills, time and project management,
- ix. Experience in leadership and contribution to projects and development of others
- x. Systematic design and evaluation of systems and processes
- xi. Demonstrated Knowledge and Skills achievement at appropriate levels (for example at a Design level on the Bloom's cognitive taxonomy scale)

2. Learning Areas

- i. Fundamentals
- ii. Resources and interactions
 - ✓ People,
 - ✓ Organizations and Society
 - ✓ **Information,**
 - ✓ **Hardware,**
 - ✓ **Software,**
 - ✓ **Communication (Networks)**
- iii. Cross-cutting concerns (security, process design, qualities, ...)
- iv. Design and Enhancement of Systems (methods, standards, artifacts)
- v. Application Domains, Issues and Topics in their context
- vi. Capstone Experiences (projects, internships, research work)

3. Policies as Enabling Mechanisms

- vii. **Institutionalization** of student research as a method of study
- viii. **Free movement** of people and ideas (no department boundaries etc.)
- ix. **Early delivery** of a broad conceptual foundations in ICER
- x. **Teamwork** for substantial process learning and study of artifacts
- xi. **Standards** of work and achievement of outcomes
- xii. **Engagement** of students assuring appropriate achievement of standards of ethical behavior and social responsibility
- xiii. **Research Faculty responsibility** for involving students in challenging study activities, relevant problem solving and assuring use of scientific methods and the achievement of demonstrable ability of students to participate in the creation of respectable results.

4. Explicitly integrate feedback from Pre and Post University experiences of students in tailoring specific programs locally.

In conclusion: I hope that a **research oriented curriculum** is to be given a proper attention in IT. It is a controversial idea mainly because of the current reputation of university 'research' as inaccessible to students, devoid of attention to real problems, and the reason for shortchanging education with two-tiered system of Teaching vs. Research.

Preparing IT Graduates for 2010 and Beyond
A White Paper

Pamela B. Lawhead

Computer and Information Science

The University of Mississippi

University, MS 38677

lawhead@cs.olemiss.edu

In my lifetime as a teacher computers have moved from being an expensive tool for high level researchers to a pervasive part of our every day lives. I have moved from being someone who teaches people how to think in a structured way and to then translate those thoughts into a computing language, to someone who must devise ways to teach an overstimulated group of young people how to survive in a world driven by vast quantities of data and controlled by embedded systems.

We need to take a step back, stop following fads, stop chasing the problem and, instead, try to review our goals. We should not be driven by industry needs but rather by the nature of the problems being addressed. If we identify first principles, and then attempt to understand the tools that students need to solve problems using first principles, we will be able to teach students to solve problems that emerge and make them immune to following fads and dead end streets. We do need to avoid the “Saber Tooth”¹ problem of teaching “fish grabbing” after the invention of the hook, but, that does not mean that we don’t still need to catch fish. In teaching programming language principles, I tell the students that when they finish the class they should answer the question “In what language do you program?” with “Any for which I understand the paradigm and have a manual.” We need to find that approach to computer science in general. For that reason, I think that the “history of computing” should become a required course in the undergraduate curriculum. While the history is short, a careful review of it finds that we keep doing things in a spiral format, always coming back to position very similar to a position previously attempted by someone else but shifted just a bit in time. If our students don’t understand that then they will be forever either following fads or reinventing the wheel.

All students should be required to have a course or two in formal logic, translating ideas into systems is a large part of what they will do with the rest of their lives, if they do not understand how to formally represent an idea then they are forever crippled.

It is important to remember that the current group of students has two characteristics never before seen in institutions of higher learning, 1) they have had the last bastion of abstraction, music, made into videos for them so their ability to abstract has been severely compromised (their songs are videos and their books are movies) and 2) they are totally connected. They have never spent a moment of their short lives very far away from a network, the internet, their cell phones, or IM to name a few. So, they need instant gratification and have little patience for the task of solving problems requiring long hours of solitude and extreme levels of abstraction. As

¹Peddiwell, Abner J., *The Saber-Tooth Curriculum, Classic Edition*, 07/27/2005, McGraw-Hill

we look forward to determining ways of teaching them, we must look to the gifts that they do have and build on those.

So, how do we prepare this unique group of students? Well, we must start with where they are, they deal with classical abstractions very poorly so, in preparing them to enter the computing workforce we must first teach them to abstract. There are many courses in the classical undergraduate curriculum that teach them to do this, we must, therefore now include those courses, not just the sciences but the liberal arts as well. We must put them back into classical philosophy, music and literature. Once the students become good thinkers, proficient with abstractions and understand their own history then they will have the maturity to capture domain knowledge and program.

Now, at the same time, this group of students deals with multiple simultaneous events very proficiently. With that in mind, we need to work with them with their strengths. They naturally deal with parallelism very quickly, they are very visual and extremely comfortable and competent in 3-D environments. This needs to be exploited when we attempt to teach them. If we pose problems to them in a story board environment then they become quite comfortable and are able to move to the implied abstractions very readily. What is sometimes disconcerting, especially as a teacher, is that we know that we would have had to work very hard attain the student's level of comfort with these 3-D, parallel areas and we presume that they must have done the same. We think that they are so sophisticated and thus we tend to overlook the fact that they not understand very fundamental language and math skills. The disconnect comes when we attempt to teach them things that require these skills, thinking that if they can build 3-D objects or text-message with one hand without looking at the key board then certainly they understand more sophisticated ideas. They do not.

The challenge for the next ten years then becomes one of recursing to levels of an earlier time, levels a little bit less sophisticated than the student's current level, teach the identified missing fundamentals and then moving forward to a marriage between the gifts of the current students and the demands of the world of pervasive computing. Students live now with our frustrations because we tend to expect more of them that the educational world has provided to them. Once they speak from a world well founded in fundamental principles they will be able to ask the questions that will arise in the future and move forward from a well-grounded position using the gifts of their current world to solve the problems of the future.

Improving Undergraduate Computer Science and Engineering Education: A White Paper Concerning Integrative Computing Education and Research

Gregory D. Peterson and Don Bouldin

Electrical & Computer Engineering

University of Tennessee

Knoxville, TN 37996-2100

[gdp, dbouldin]@tennessee.edu

Abstract

Advances in technology and changing economic conditions due to globalization present profound challenges and opportunities to computing education. The rapidly changing landscape for computing makes it necessary to update the topics in undergraduate curricula. The same technological advances present exciting opportunities in developing adaptive, individualized pedagogical tools.

Introduction: Challenges Facing Computing Educators

A number of trends dramatically impact the educational landscape, particularly with respect to computing. Technology continues to move ahead at a tremendous pace (e.g., Moore's Law for semiconductors), resulting in constantly shifting skills and expertise. Inflection points have been reached or are imminent across a spectrum of computing topics:

1. Computer architectures are no longer bound solely to the von Neumann paradigm as new architectures and computational models become available. Examples include quantum computing, nano/biocomputing, configurable processors and reconfigurable computing.
2. Device technologies based on CMOS with silicon may not be able to continue their historic growth as predicted by Moore's Law. New devices, circuits, design tools, and methods may be required for developing microelectronic computing systems.
3. Materials for computing may soon shift to bio/nano devices, with profoundly different characteristics for design, manufacturing, timing, power, and behavior.
4. Networking is shifting from wired to wireless to ad hoc networks, with resultant changes to protocols and distributed algorithms.
5. Networked, embedded devices will become ubiquitous, resulting in an explosion in the number of computing devices and their potential interactions.
6. Mobile, robust code (e.g., agents) are becoming popular to enable searches, distributed control, or similar applications.
7. Distributed peer to peer networks, data, and applications remove centralized control, with implications for resource allocation, intellectual property, and governmental control.
8. Open source, collaborative development are becoming more of the norm, potentially with ad hoc, self-organized development teams of geographically distributed, diverse skills.
9. Broader societal issues are becoming more prevalent (e.g. security, privacy, freedom of speech, crime, infrastructure protection).

In addition to the technological advances related to computing, economic and social changes resulting from globalization must be addressed.

1. Employees (engineers, programmers, etc) must provide value or their jobs will be eliminated.
2. Employees can come from any region, ethnicity, gender, or background, so long as they are productive.
3. Employees must have the ability to work with diverse, geographically distributed teams.
4. Virtual laboratories or design centers for scientific discovery and product development enable accelerated advances while reducing dependence on capital equipment.

Taken individually, each of these points would present interesting pedagogical challenges to computing educators. Together, they illustrate the tremendous challenge facing educators that strive to prepare undergraduate students for the workforce of the 21st century. The educational system in the U.S. is inadequate in preparing the workforce to fully participate in the global economy. In the primary and

secondary schools, students are not provided enough background in math and science to enable success in computer science and engineering. All levels of education need to be more effective in teaching skills related to computing. We need to ensure that future ideas about undergraduate education are not too limited by technological anachronisms.

Opportunities for Improvement

The traditional method of teaching based on a lecturer addressing an audience of students is outmoded and inefficient. Politicians and clergy have transformed their presentations from structured speeches or sermons (e.g. three points and a poem) to multimedia presentations. These newer forms of communication better reach newer generations of listeners raised in a post-literate society driven by rapidly changing cinematic/television scenes. Computing educators need to embrace the collection of technologies that adapt to enable the students to learn based on their particular preferences and capabilities. The powerful, networked computing systems provide the potential for adaptive and cooperative learning that can tap into the vast resources of the web.

As a nation, we cannot afford to lose students during their undergraduate studies. The ability to learn at one's own pace and to explore various topics in more detail should help with retention. Better marketing of the field to potential students and engaging current students in exciting projects that include broader issues should help as well. More internships and "real world" experiences should be sought within courses and the curriculum to provide motivation for the studies and to reinforce learning.

Collaborative development should be pursued to better prepare students for current design practice. For example, at UT we have teams of students develop portions of behavior for SoC designs. Other universities can build on this infrastructure as well. Collaboration between students in different regions or countries can then be aggressively pursued to allow each group to take part in a much more challenging design than could be pursued otherwise.

New technologies provide exciting opportunities for outreach. Computing educators need to make projects more fun and approachable to reach students beyond computer science and engineering. For example, using cellular automata or game of life types of exercises in logic and exploring emergent behaviors are intuitive and students like them. Similar topics could be adapted for primary or secondary students in math, science, social studies, etc. Simulations of behavior (including applications similar to Sim City) can also be used to engage students. Changing the rules of behavior can then be explored to see what happens as an exercise in anthropology or political science. For computing students, more exciting ("fun") projects should be employed and publicized. For example, students could program robots or virtual players (e.g., soccer or football players in a game between student programmed player teams) with a highly visible playoff during the Superbowl/March Madness/related sporting event.

NSF/Government Role

Government funding of research into advanced pedagogical techniques, tools, and infrastructure is needed. Specific focus should be given to research that enables adaptive, individualized teaching and learning, particularly for bridging cultural differences. (Why isn't there a high quality, adaptive teaching module/web site for calculus or related topics to enable students from all cultural and economic backgrounds to learn the necessary skills for success in computer science and engineering?) For advanced topics, particularly for skills of high value in the job market, the free market system should be effective in creating and extending an efficient set of such tools and modules, although the government will need to seed the development for this industry. For primary and secondary schools, government funding will be essential for the development of a spectrum of educational tools and curricula that span the necessary ages and topics. This funding could come from federal, state, or local sources.

References

Computing Curricula for Computer Engineering Joint Task Force, "Computer Engineering 2005: Curricular Guidelines for Undergraduate Degree Programs in Computer Engineering." (draft) October 2004.

1. *What are the biggest challenges that ETSU's CIS Dept. faces in undergraduate education?*
 - a. *obtaining enough resources to maintain enrollments and recruit new students, while meeting increases in professional expectations.* We offer CS and IS concentrations; added an IT concentration in 1999; and, still, our major classes are down 60% from 2001. In an effort to attract students, we're now adding emphases in security and game programming. In the meantime, we've lost a faculty line; may lose a second; and are laboring under the new, heavyweight CSAB/ABET standards for accreditation and IS/IT/CS differentiation.
 - b. *coping with the perception that outsourcing has made computing a dead-end profession in America.*
 - c. *addressing the lack of female computing majors.* Our percentages of new, incoming females in our fall 2005 undergrad and grad programs were 5% and 0%, respectively. The only bright spot for most of our students is that we're about to co-locate with the college of nursing ☺.
 - d. *coping with the decline in basic reading, writing, and thinking skills amongst our incoming students.*
 - e. *teaching students to read and write well enough to be competent computing professionals.* See also d.
 - f. *finding room in a liberal arts curriculum to adequately train CS students.* Employers hire our CS graduates because they're skilled in software development. However, our students, as a rule, need five four-credit programming courses (CS Ia, CS Ib, data structures, file structures, and assembler) and two semesters of software engineering to obtain these skills. When you add in the obligatory systems courses for us, operating systems, computer org, networking, and database, and piggyback this onto the general education curriculum, you're close to the state-mandated 120 credit-hour maximum. We also teach ethics throughout the curriculum. As a result, we don't have much room for teaching like problem decomposition and solving, software maintenance, testing, scripting---let alone programming languages, algorithms, parallel programming, professional issues, scientific programming, embedded systems, NA, AI, and theory.
Our other concentrations face similar issues, relative to concerns like HIPPA and FERPA (IS) and currency with emerging technologies and hardware maintenance (IT).
 - g. *encouraging students to think creatively about computing as a profession—e.g., to engage in research, and to do internships.* This concern is addressed, to some extent, by our capstone experience course in our IT concentration. The CS and IS concentrations, unfortunately, don't have the same room as the IT.
 - h. *threat of faculty burnout.*
2. *What might the community do to address the challenges you identified above?*

To paraphrase Warren Zevon, forget the lawyers and guns; send money ☺. Here are a few ideas for using additional resources to buttress undergrad education:

- a. *start computing education sooner.* Encourage high schools to offer courses like programming and discrete math, and to send more gifted students in grades 9-12 to community colleges and colleges for training. This would take closer cooperation between K-12 and post-12 educators, and possibly additional training for HS instructors, including help with building these courses. The time for this has to come from somewhere.
- b. *alternatively, lobby for, and move to, a five-year model for CS education.* This would require a change in the political climate, and maybe a joint effort with other engineering professions. But, maybe, the time scale has already lengthened, given the increasing importance of the MS in the computing fields.
- c. *educate America about the value and continuing importance of computing.* Speaking to questions of value might help to address the perception problem, along with the female majors problem—if, indeed, American women tend to focus more on the uses of technologies, rather than the technologies proper (cf. Why Gender Matters, Leonard Sax, Doubleday). One member of our faculty suggested that a prime-time, computing-oriented equivalent of CSI might help ☺.
- d. *provide more scholarships and summer work opportunities to undergraduates.* Another reason that many undergrads do "little" outside of class is that they must (or feel that they must) work 20-40 hours a week to help meet expenses, while taking full course loads (to qualify for loans).
- e. *organize more regional low-cost workshops that allow undergraduates to witness research in action, and to get involved in it.* Here, I don't mean refereed, multi-day affairs like ACM regionals, but, rather, *informal*, no-frills, one-day workshops, designed primarily for BS, MS, and Ph.D. students in computing. I organized such one workshop in the mid-Atlantic states that's lasted ten years (cf. <http://www.research.ibm.com/masplas>), and am currently organizing another for southern Appalachia (cf. <http://www-cs.etsu.edu/sasplas>). You can find a paper on a model for these workshops at <http://www.research.ibm.com/people/h/hind/97-108.ps> (co-authored with Mike Hind of IBM). Such workshops also cost money, if you want to waive admission (which helps), and buy release time for the organizers (which I've never seen happen).

- f. *raise the standards for written work and critical thinking throughout the curriculum, beginning with our own classes.* This will take lots of close reading, and frank, hard, and careful critiques of the crap that often passes for technical writing in homework assignments (e.g., SRSEs and documentation) and term papers. It might also mean (re)training in technical writing for computing instructors.

The argument that money would help is probably quixotic, given the impact on government budgets from Hurricanes Afghanistan, Iraq, Katrina, and Rita. But you asked ☺.

3. *What might an ideal undergraduate model for computing education look like in five years?*

In April 1993, I heard a lecture on a similar concern at Yale's Alan Perlis Memorial Computing Symposium: a talk by Peter Naur on the 1960's-era quest for the Ideal Programming Language. After explaining the rationale for the vision—a hope for a language that would help programmers to Do Right—Naur gave his reason for the quest's failure: the existence of irreconcilable differences in how people grasp ideas. I draw similar conclusions from the literature on learning styles, on project management, and, unfortunately, on education.

Now, would someone care to rephrase this question? ☺

4. *Can you identify inhibitors that might prevent the nation from achieving goals it sets for computing education?*

I mentioned a lack of resources in 1. Other concerns include the lack of a clear rationale for why computing still matters and why more majors are needed. Beyond this, where in America today can we find a champion for science education like JFK?

Can you identify strategies that may transform undergraduate computing education in the USA?

First, identify good reasons to study computing, and promote them. During the 1950's and 1960's, fear and the quest for a better life helped to drive the national investment in education. Unfortunately, the focus on science that began with Sputnik ended with the collapse of the Evil Empire. And, as for the quest for a better life, Mike Royko wrote a telling column during the 1980's in which his alter ego, Slats Grobnik, said that America's #1 problem was too much Stuff. "Kids", said Grobnik, "learn their VCRs before their ABCs."

I would hope that a vision of computing as a profession that allows people to earn a good living while creatively serving others could replace the old motivators. We also need to say that the 1) the most desirable positions are not outsourced; 2) these positions are currently being filled by baby-boomers, who 3) will soon retire, leaving lots of opportunity for well-prepared, highly motivated individuals—the kinds of student we need to attract.

If this message seems reasonable, then it needs to be sold in elementary and high schools, using partnerships between colleges and school districts, to the extent that this is doable in this imperfect world. I fear that people who teach K-12 are embroiled in their own nasty set of problems, and might not have the time to listen. Still, it would be good to try, all the same—in part, by beginning by listening to their problems, and seeing what we can do to help, in this same spirit of service.

Our department currently supports two goodwill-building, outreach efforts for local K-12 students. One, PASTA (Providing Area Schools with Technological Assistance), sends surplus computers to elementary, middle, and high schools, together with senior IT students to help with installation, training, and support. The other, GIST (Girls In Science and Technology), hosts a series of three summer computing camps for 10-, 11-, and 12-year old girls, respectively.

In the meantime, we have finally started, as of this year, to conduct recruiting visits to area schools. This, of course, takes even more time and resources.

A colleague also suggested more co-op work as a way of supplementing what people are learning in school. As Michaele puts it, "Mandatory co-op work (back to 2d) would go a long way to help students teach themselves and understand/appreciate what they are getting in the classroom."

5. *What stakeholders should be involved in designing strategies to catalyze the transformation of university computing education throughout the nation?*

Whatever stakeholders get involved, the watchword should be, "First, do no harm". I fear grand initiatives because those who promote these tend to produce grand-sounding but imperfect, even wrongheaded, fixes for educational quandaries: e.g., the "new math", the abandonment of grammar, and, more recently, "no child left behind". I hope people are wise enough to apply Frederick Brooks's insight—"No Silver Bullet"—to the current "crisis" in computing education, and to be content to work steadily to find effective, if generally unspectacular, strategies for achieving long-term progress.

ICER Workshop White Paper

Han Reichgelt, College of Information Technology, Georgia Southern University

Challenges. To my mind, the greatest challenge facing computing education over the next few years is the ability to attract high ability students into the field. Information Systems has already seen a precipitous drop in the number of undergraduate students. One suspects that an important reason for this is the fact that many students who might potentially be attracted to a career in computing are put off by the incessant media campaign against overseas outsourcing¹ of manufacturing and information technology operations. Such campaigns usually focus on the number of jobs that have been lost and ignore the number of typically higher paying jobs that have been created in this country because of globalization. While such noises may have little effect on those potential students who in the words of Margolis and Fisher (2002) feel a “magnetic attraction” to the computer, it is likely to affect those who are more interested in “computing with a purpose.” The much more rapid drop in enrollment in IS programs, compared to the enrollment in other computing programs, fits this pattern. Students who might be interested in a program in Information Systems are more likely to be attracted to a more applied form of computing, and may therefore be more interested in an IS program that covers “computing with a purpose” than a more theoretically focused programs in Computer Science.

Reacting to the challenge. There are two ways for the computing education community to deal with the phenomenon of overseas outsourcing and the concomitant “disappearing students” problem. First, the community can lobby the public or government to somehow ban overseas outsourcing. Second, the community can accept overseas outsourcing as a fact of life and redesign its curricula to prepare students for those careers that are harder to outsource.

I regard the first reaction as misguided. Any jobs that may be saved in through lobbying efforts of this type are likely to result in increase inefficiencies in the overall economy. Thus, while decisions by some local and state governments to have their call center operations performed by companies based in the United States using US labor may have saved some jobs, the increased cost of running of these operations amounts to a heavy public subsidy of such jobs. In general, subsidies of this kind have proven to be an inefficient way of spending public money and by distorting international trade have helped keep poor countries poor, an argument which is now widely accepted when it comes to farm subsidies.

Given my skepticism about the viability or desirability of the first reaction, it will come as no surprise that I strongly believe that the computing education community should accept overseas outsourcing as a fact of life, and should redesign its curricula to

¹ Most of the media fail to distinguish between *outsourcing* and *overseas outsourcing*. Outsourcing of jobs by organizations is a well established and widely accepted practice that allows organizations to concentrate on their core competencies and reduce cost. Very few organizations run their own office cleaning operations and many hire external logistics companies to run their logistics operations. In general, the media do not object to outsourcing if it increases the operational efficiency of an organization and leads to the establishment and growth of outsourcing organizations, as long as these organizations are US based.

prepare students for those careers that are harder to outsource overseas. Clearly, this can only be done after a thorough and wide-ranging discussion on what computing jobs are hard to outsource overseas. This discussion should involve not only the computing education community but also organizations that employ a large number of IT professionals, including those that have outsourced some of their IT operations overseas. It is also important that the results of these discussions are widely disseminated, and in particular, that potential students are made aware of the results of these discussions.

Models for Transforming Computing Education. While it is presumptuous for me to pretend to know the answer to the question which IT related jobs are hard to outsource overseas, I offer the preliminary suggestion that attempts to outsource the following two types of jobs overseas might lead to problems:

1. Jobs that require high levels of security;
2. Jobs that require the IT professional to develop an intimate knowledge of the organization.

If the above analysis does indeed stand up, then there are two potential consequences for computing education. First, information assurance and security has to be given a higher level profile in computing education than it currently has and must be woven throughout the curriculum, rather than be the subject of a few isolated courses. The recently draft IT Model Curriculum (www.acm.org/curriculum) might be a good place to start for this discussion.

Second, in order for computing students to be prepared for the second type of job, they must be given the opportunity to develop the skills that allows them to gain intimate knowledge of organizations and communicate with members in those organizations. It is ironic that students enrolled in IS programs are more likely to develop these skills than students enrolled in many other computing programs. However, it is also important to recognize that computing is used in many more areas than just business. It might therefore be good if computing programs allowed students the leeway to develop in-depth knowledge of some area in which computing is likely to prove useful, whether this is the administration of justice, business, education, military science, music or policy making. The current ABET CAC accreditation criteria for Information Systems explicitly mention the inclusion of an “IS environment” in the curriculum criterion, while the ABET EAC accreditation criteria for Software Engineering mention the “ability to work in one or more significant application domains.” Both, it seems to me, point in the right direction.

Margolis, J. & Fisher, A. (2002) *Unlocking the Clubhouse: Women in Computing*.
Cambridge, Mass: MIT Press

*Integrative Computing Education & Research (ICER)
Preparing IT Graduates for 2010 and Beyond*

Rebecca H. Rutherford
Southern Polytechnic State University
Marietta, GA 30060
brutherf@spsu.edu

1. Preparing undergraduates for computing careers: as an educator, I find that there are several challenges for myself with students majoring in information technology, computer science and software engineering. First, the students still come to college prepared with a variety of computing skills – from general “WORD” knowledge, to having programmed JAVA in high school. It is still hard to have beginning programming classes with such a variety of skill levels. Second, students seem to be technology “savvy” – in other words, they have their technology toys (iPods, etc.), but don’t really understand what the world of computing work entails. One of the biggest challenges is to get the students to “learn how to learn” with our dynamically changing computing environments.
2. Transforming the educational experience: companies can help with initiating contacts with local educational institutions. Normally, a faculty member has to try to go out to a company to get speakers, projects, etc. It would be a much appreciated thing if companies would seek out and foster relationships with education.
3. Models for transforming computing education: if we take a more multi-disciplinary approach, I believe such a curriculum will need various “tracks” of study. There still should be some basic computing skills that would be needed – programming, op systems, basic architecture, database – but after that, I believe that we will also need to think more seriously about how the world is changing – the web, networks, security, business domain knowledge. A curriculum can’t be everything to everyone, but we have spent so much time trying to identify the differences in CS, IT, SWE, IS and EE that it will now be difficult to come up with a generalized computing model.
4. Inhibitors and strategies: first of all, business needs to understand that when they ask for job applicants – they are usually advertising for “skills” – “We need a JAVA programmer who has worked for 5 years”. I understand where companies are coming from – they need someone to get to work immediately, however, with technology and computing changing so rapidly, it is very difficult for applicants to have lots of experience in a particular thing, and moreover, new graduates may not have the type of “skill” companies are looking for. Companies need to adjust their mindset and learn that computing graduates are educated in computing basics and that they have the ability to “learn” and learn quickly – adjusting to changing technologies. Second, since schools are in a competition game (for students), it is hard to get much cooperation between entities. National accrediting agencies (such as ABET) help to standardize curriculum guidelines, but each school puts their own slant on their particular curriculum. Schools would do better to cooperate and participate in shared degrees. Third, we have

- almost no Information Technology doctorate degrees at this time – most are CS or IS. We have broadened undergraduate majors (CS, SWE, IS, IT, EE, etc.), yet have not kept up with creating faculty who have credentials in these areas. If we think of a more consolidated approach to a computing model, then we will also need to revisit what constitutes teaching in these areas – does a multi-disciplinary faculty make sense (computing, IT, SWE, EE and business)?
5. Who might participate: it will take a lot of “persuasion” to drive the computing education domains into a more multi-disciplinary cooperative unit. First, it has always been a “bone of contention” as to whether companies should drive the curriculum, or that education and pedagogy should drive the curriculum. Educators realize that there is great merit in Education for the sake of Education, but also recognize, especially in the computing fields, that we are educating our students to work. Both large and small colleges/universities as well as various companies, large and small, accreditation personnel, and government personnel should all be part of such an initiative. The process would be similar to what Information Technology has done over the past two years with establishing a model curriculum and accreditation guidelines.

ICER Workshop White Paper

Judith L. Solano, Chairperson
Department of Computer & Information Sciences
University of North Florida
Jacksonville, Florida

The biggest challenge we currently face in education is capturing the attention of potential students and convincing them to choose a career path in the computing sciences. With the demise of the dot-coms many who saw computing as a quick avenue to lucrative careers lost interest. Other sobering economic trends that have had a chilling effect on enrollments include downsizing and outsourcing, both of which have received a great deal of attention in the media. Students, and maybe more importantly their parents who will be paying their educational bills, are fearful that studying the computing sciences will not result in gainful employment upon graduation. As a consequence, they have been migrating toward other majors.

It is fairly commonly accepted that economic trends are cyclical. It would be reasonable to expect, therefore, that at some point jobs would return and students would follow. In fact, the data now seems to suggest that the job market has begun to improve, but the students have not yet followed. Perhaps it is still too soon for parents and their children to be confident in an improving market and if we wait patiently, we will eventually see improving enrollments. Disturbingly, today's teenagers seem to suggest otherwise.

We now seem to be faced with a much more insidious challenge. Today's teenagers do not seem to be interested in studying the computing sciences. They are already very sophisticated and talented users of the technology that pervades almost every aspect of their lives. They don't see it as anything that is particularly new and interesting. They know they don't really have to know and understand it to use it, so it doesn't really present a challenge. And, since it has always been available to them, they haven't given the first thought to who is going to enhance it and who is going to develop the technology of the future. As educators, we are faced with capturing the imagination of a generation of students with whom we are out of touch.

A first step in any strategy to address our challenges would be for us to get to know today's students. All texts on effective presentation skills stress the importance of knowing your audience. Some of our greatest military leaders have written and spoken of the importance of knowing your enemy. We need to know our students and to do so we probably need to enlist the aid of professionals trained and skilled in analyzing human behavior. As we embark on this endeavor, we must be careful not to focus all of our attention only on undergraduate baccalaureate candidates. Many are suggesting our efforts need to begin in the high schools. Time Magazine's recent issue on "Being 13" gives one reason to believe the middle schools would be an even better starting point.

An important next step would be to launch an advertising campaign designed to change the face of computing. We must get the attention of our audience, pique their curiosity

and imagination, and then reel them in; much like corporate America does when it launches a new or redesigned product. A hit TV show like CSI wouldn't hurt either, but that may be too much to hope for.

We can't forget the importance of parents in the lives of their children. They are now being referred to as "helicopter parents," because they remain hovering over their children well into their college years and beyond. They are heavily invested in every aspect of their children's lives. If we hope to get the children interested in computing as a field of study and a possible career, we must regain the confidence of the parents. They have the perception that all of the computing jobs are in India or China. They need to believe that there are and will continue to be employment opportunities for their children in the U.S.

If we can get the attention of students and their parents, we then need to provide an educational experience that will retain their interest. It is clear this will require curricular change and change in the delivery methods we employ. But, exactly what this change must be and what computing education should look like in the future is not so clear. Recently, Bill Gates declared we need to be innovative, but he had no suggestions on what we might do that would be innovative. This is probably because he, like so many of us, does not truly know and understand today's student – our audience. This is a critical task for us, upon which everything else will depend.

The new curriculum is likely to need to be multi-sensory, very fast paced, and interactive. Today's students after all are the ones who popularized extreme sports and brought us the X Games. They have little patience, a lot of energy, and are willing to take great risks. The fact that they do not want to learn the notes and the scales, for example, and they do not want to practice is telling. They just want to be able to pick up an instrument and play. Their approach to computing and technology is no different. They want to get their hands on the technology and start using it. They do not read instruction manuals. The educational experience can no longer be passive. It will have to be very active, if we are to have any hope of retaining our students.

Another likely characteristic of the curriculum is that it will need to incorporate a multitude of disciplines. Today's students take the technology for granted. They are not necessarily interested in the technology for its own sake. They are interested in what they can do with the technology or, said in another way, what the technology can do for them. Currently we tend to teach the fundamentals of the computing sciences independent of developments in health care, in education, in criminal justice, or in music and the arts, etc. The challenge will be to provide an environment wherein students can focus on those areas of their lives that they find most interesting and engaging, while integrating the fundamentals of computing. We want them, almost without realizing it, to gain the knowledge and skills that will enable them to enhance the technology they now take for granted and even to imagine and develop the new technology of the future.

ICER White Paper
Pradip K Srimani

Computing has become ubiquitous in our society, and it now affects countless aspects of our daily lives. Indeed, it is difficult to imagine any aspect of life in the modern world that is not affected in some significant way by computing. Applications of computing, once restricted to a narrow range of disciplines within science and engineering now include agriculture, athletics, arts, business, health care, recreation, government, military, and the social sciences, in addition to all traditional science and engineering disciplines. It is thus no surprise that the effective use of computing can be found in all academic disciplines of any College or University.

To paraphrase [Jim Foley from his CRA position paper](#), – whereas “science” discovers the laws of nature and “engineering” uses those laws of nature to create physical artifacts, “computing” discovers and uses the laws of how to compute and how to organize information to create computational and information artifacts to solve problems in any application domain more efficiently. Computing is a unique discipline where the academic and the professional and application aspects are very intricately intertwined.

1. *Preparing undergraduates for computing careers:* Current and future students who would seek careers in computing continue to need a core knowledge of the fundamentals of hardware and software system design, but such core knowledge is no longer enough. Deep knowledge of the domain of application of computing is almost always a co-requisite for effective problem-solving and discovery. Over the last several decades important contributions have been made to design automation, operating systems, compiler construction, networking and modeling. However, those who study computer science to pursue a career outside the academy (the overwhelming majority of students) often find that what they learned in computer science is important to the effective application of computers, but they must learn other domains (finance, architecture, insurance, civil engineering, construction). Likewise, because we have made computers more accessible, professionals in other disciplines are using computers in their research and exposing their students to computer applications, often with less than effective results because they lack the foundation that computer science offers. While core computer science remains an important discipline, we believe we are missing significant opportunities to contribute more broadly to our society and better prepare our students for leadership roles in that society.

We need to prepare students to meet the information technology needs of business, government, healthcare, schools, and other kinds of organizations. There are numerous applications areas that offer significant long term challenges requiring multidisciplinary teams composed of traditional computer scientists and professionals in other disciplines to develop the infrastructure (equipment, software, information) that will facilitate progress toward addressing the challenges. As these intersections broaden, more training in the intersections is required in order to function more effectively therein; problems unique to these intersections will emerge, requiring the understanding and study of new underlying principles of computing in these domains. *What is needed therefore is the development of application specific degree programs; which are focused more directly on the central aspects of these intersections, and away from the central areas of traditional computer science.*

2. *Transforming the educational experience:* Traditional computer science educational experiences are notoriously defined by the countless hours students spend out of class developing software, writing, debugging and testing computer programs. Almost no time is spent studying, analyzing, or critiquing their effectiveness. But these concerns are foremost to users of computing in the application domains. In the future more training will be needed in the effectiveness of application software and software systems, and in making effective use of increasingly sophisticated software systems. We will need increasingly smart users, who know how to get the most out of existing software systems, who will know and understand the limits of these systems, and who will know what is needed to make these systems more effective and beneficial. Thus future computer

science courses will be less focused on training in the fundamentals required for producing future systems and more focused on the effective utilization of existing systems; in order to produce more efficient users.

3. *Models for transforming computing education:* It is expected that the diversity of backgrounds and interests across the entire spectrum of computing education would be enormous, probably a microcosm of the entire comprehensive university itself. Still, each student, irrespective of his/her area of specialization must have a solid grounding in the fundamentals in computing – technologies (hardware & software), impacts on society, networking, security impacts, pervasiveness, legal, ethical, and privacy issues. After the core, students should choose specialized tracks, and curricula for each track should be designed separately. Here is a list of possible (in no way an exhaustive list) tracks: (1) computational arts – computer graphics, visualization, digital animation, film, image processing, virtual reality, human-computer interaction, game development, multimedia systems, computational aesthetics (2) Computational Science – bioinformatics, computational geometry, mathematical modeling, quantum computing, systems simulation, optimization, computational geography, computational astronomy (3) Computer Science – computer architecture, software engineering, operating systems, languages, theory of computation, security, networks, embedded systems, sensor systems (4) Information Systems – database management, systems analysis, requirements analysis, infrastructure design and management, e-commerce, operations research, enterprise architecture; (5) Computer Engineering – digital systems, IC technology, VLSI technology, signal processing, system integration. As new disciplines emerge through the integration of computing with application areas, those trained in the computing disciplines have an opportunity and responsibility to lead. It is also to be noted that the tracks are not mutually exclusive, there will be significant overlaps; both the students and the faculty should be able to share and exchange capabilities and resources.
4. *Inhibitors and strategies:* Traditional college structures in the universities put significant constraints on the seamless integration of courses to form the needed curricula for specific needs; there is a lack of mechanisms for inter-college collaboration in both teaching and research and creative activities. For example, faculty in science and engineering are in general evaluated in terms of research publications and external funding whereas faculty in performing arts, fine arts and business disciplines are evaluated in terms of other metrics. One possible strategy, which is being tried on several campuses, is to establish a College/School of computing in the broadest sense of the term. Curriculum, hiring, tenure, and retention decisions should be effectively made with the diverse nature of the discipline in mind.
5. *Who might participate:* There are many stakeholders. Professional societies should develop new curriculum models to accommodate diverse needs for diverse kinds of computing professionals and educators and researchers with computing at their cores. Universities should provide computing as part of General Education to all majors, come up with appropriate mechanisms to encourage collaborative research and education; reward mechanisms should be appropriately and flexibly tailored (faculty in science & engineering and faculty in fine arts or performing arts are not evaluated by the same metrics; how to meaningfully evaluate faculty engaged in research and education in digital production arts for example?) Government, especially federal funding agencies should fund creative endeavors to investigate possible solutions, create new curricula, run pilot programs, and raise awareness about the need for computing basics. K-12 programs should be looked into to incorporate computing education at that level, provide proper training for high school teachers, organize training workshops, and to encourage and search for computing talents from the very beginning.

Active Learning

Craig Thompson

Professor and Axiom Database Chair in Engineering
cwt@uark.edu, University of Arkansas, Fayetteville, AR 72701

Thesis: to some extent, undergraduate education is founded on widely assumed myths that are built into the way we educate. What could change if we recognize problems with these assumptions?

Myth #1: Undergraduate advising should be focused mainly on what classes students should take.¹

Reality: Advising and courses should provide students with an understanding of the scope of computing, how it relates to nearby disciplines, and the wide range of applications, including cross-application disciplines. Campus culture and foreign travel should be encouraged early on. Interning, cooping and work experience related to the major should be strongly encouraged from sophomore year on. Some companies will not hire without this.

Myth #2: Undergraduates should take classes and read text books. **Reality:** While books do encapsulate a core of what is worth knowing and hint at the shape of a field, many undergraduates get all the way through four years of school without ever reading a research article, knowing about online ACM and IEEE, or attending a conference. So they do not know about the rest of the iceberg of rich resources they could use to solve problems.

Myth #3: A senior-level Capstone projects course is enough to teach undergraduates about projects. Teams, projects, reinforcing software engineering, and presentation skills should be the Capstone focus.

Reality: While good, this is not enough. Project courses should start at the sophomore or junior level. There is a need to teach career skills, how to start a small business, how to bid a contract, how to apply for a patent, how to think outside-the-box or comfort zone, etc.

Myth #4: Undergraduate students should not take graduate classes. **Reality:** Undergraduates with prerequisite knowledge should be encouraged to take graduate classes and vice versa.

Myth #5: Computing undergraduates mainly need to know how to compute. **Reality:** Almost all I ever do is write or present. I was not taught to teach. Our industrial advisory board faults all of engineering for turning out students who know little about business. Our undergraduates never admit to having a knowledge of contemporary issues.

Myth #6: Its OK to teach software engineering without teaching computational tools. In general, it is better to teach undergraduates how to program and then they can learn any language or tool. **Reality:** Nearly no faculty nor students in computer science or computer engineering know how to use MS Word very effectively (e.g. Document Map, styles, revision mode). Some do not know what firewalls do or how to backup their machines, even how to build web pages. Tools like CVS, Javadocs, project management, IDEs, UML, XML, SOAP, ... are not always part of courses. At least some of these should be included into courses if not taught.

Myth #7: All student projects are toy world projects. **Reality:** Students projects *can* matter, can lead to results that industry uses, can lead to inventions, small businesses, and industry standards. It is empowering to believe this.

Myth #8: Research is for graduate students. **Reality:** In industry, people are asked to solve problems. So undergraduates need to learn how to frame problems as well as solve them. Also, in industry problems do not have well defined scope so learning how to divide and conquer is important. Mixing undergraduates and graduates in problem solving is good for everyone.

Myth #9: We have succeeded when our students graduate. **Reality:** We have succeeded if our students graduate and already have a job or graduate school lined up. I have asked too many students, "So,

¹ As an undergraduate at Stanford, I graduated with a math B.S. and did not know what mathematicians do – that has always bothered me.

graduating next week – what’s next?” “Oh, I guess I’ll get a job.” That is not the right answer. A metric of success should be, how many walk with a job offer in hand. How many programs actively track this metric or take some responsibility for launching careers?

Myth #10: A B.S. is a good place to stop – its time to get a job. **Reality:** The M.S. degree should be the termination for computing professionals. We should strongly encourage our undergraduates to think in terms of careers, not jobs, and M.S. rather than B.S.

Myth #11: Accreditation is the answer. **Reality:** Accreditation tends to make our courses more uniform and introduces continuous improvement, never bad. But it does not really encourage radical approaches to education reform. We should not be afraid to experiment with new approaches.

Myth #12: Isolation is good - professors should not collaborate with each other or with those in other departments or colleges. **Reality:** When students collaborate with professors (plural) in joint projects, everybody wins.

Myth #13: Faculty teach, students learn. **Reality:** Look for ways to use your students so they learn and you can learn from them. Faculty can use special projects classes to explore areas they want to know more about. Even term papers on topics they want to know about can accomplish their life long learning goal. Students get experience through this role reversal; faculty stay current or increase their breadth.

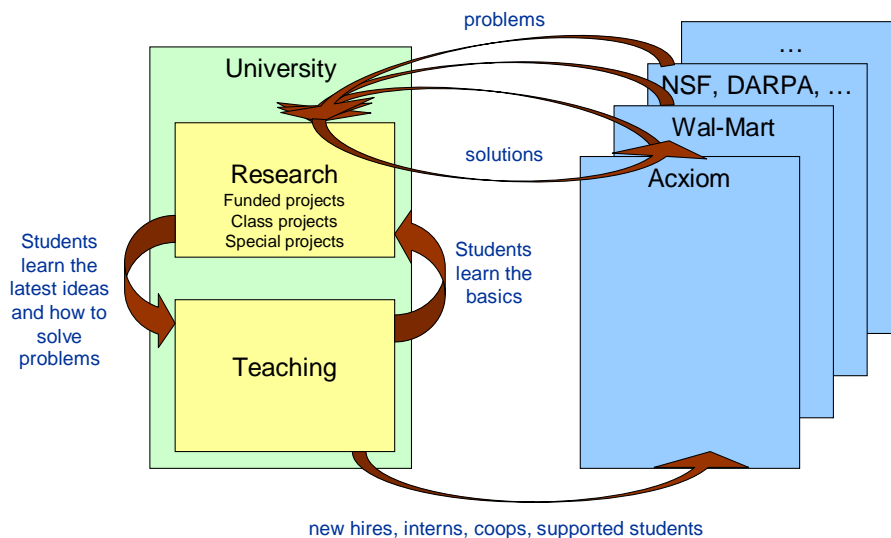
Myth #14: A faculty member should spend 40% teaching, 40% research, and 20% service. **Reality:** A triple threat is when these align. That means teaching must align with research. See Figure 1.

Myth #15: NSF funding is the name of the game. **Reality:** While NSF funding is prestigious, solving real industry problems can ground a research program and lead to results that will be used.

x x x

Active learning is learning that engages the learner. Problem solving in mixed teams of faculty, graduates, and undergraduates, engaged in solving problems for industry is a good form of active learning. The requirement for this form of active learning could be 12 – 18 hours instead of the usual four in Capstone. More would be radical. Maybe a whole curriculum structured this way could work?

Figure 1: Knowledge Cycle



Appendix D: Invited Speaker Presentations

Educating the Next Generation of Computer Science Students

Issues and Challenges

Two challenges

- n Convincing high school students to want to study CS
- n Keeping college-aged students interested in studying CS
 - n What we teach
 - n How we teach

Doomsday statistics

Percentage of incoming undergraduates indicating they will major in computer science

- n declined by more than 60% between the fall of 2000 and the fall of 2004 to just 1.5%
- n is now 70% lower than it was during its peak in the early 1980s
- n among women fell 80% between 1998 and 2004, and 93% since its peak in 1982

from CRA and Higher Education Research Institute at UCLA

October 27, 2005

ICER

Job outlook

From 2002-2012 of fastest growing jobs

10 of 20 \pm healthcare

5 of 20 \pm computer occupations

Networking and data communications analysts

Software engineers – applications and systems

Database administrators

Systems analysts

by for Bachelor's Degrees

Bureau of Labor Statistics

October 27, 2005

ICER

A High School Problem?

A recent survey by the Organization for Economic Cooperation and Development ranked 15-year-olds from the United States as 24th in math out of 29 industrialized countries. As if that were not bad enough, their science skills were even worse.

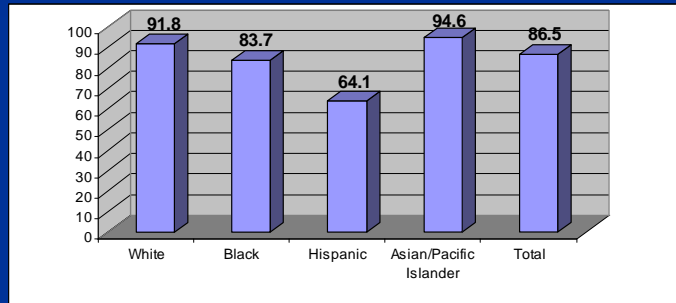
Bureau of Labor Statistics

California

- n Half of all minorities drop out of high school
- n 20% of all students do not pass the high school graduation exam

Nationally

n percent of 18- through 24-year-olds who were high school completers in 2000



October 27, 2005

ICER

IBM Foundation

“Over a quarter million math and science teachers are needed ... that is like a ticking time bomb, not just for technology companies, but for business and the U.S. economy.”

Offer to subsidize employees who leave IBM to become school teachers
Stanley Litow, IBM Foundation head



Newsweek, Sept. 26, 2005

October 27, 2005

ICER

Newton Fellowship Program

OVERVIEW



- focuses on the shortage of adequately qualified mathematics teachers in public high schools
- trains mathematically-talented individuals to become high school math teachers
- supports them in the early years of their careers
- currently operates in New York City
- will appoint over 180 Newton Fellows in NYC between 2004 and 2008
- full tuition scholarship for Masters degree
- \$90K stipend over first 5 years



October 27, 2005

ICER

Gates at UW-Madison



Gates shared real-life reflections on the promise of the technology industry.

October 27, 2005

ICER

When the blogs say...

- n “Personally, I hope my kids never get into IT; I hope they become the driving force within any company that IT works for.”
 - n “I would NOT recommend that my children look for an IT career. I would recommend that my children look for skills and an occupation that can last them a lifetime (40+ years) and not be stolen away from them by a cheaper worker/industry changes. An occupation where they become more valued and recognized as the older and more experienced they get.”
-

October 27, 2005

ICER

More bloggers...

- n “The thought of IT as a career is long over. Why? Because a career is supposed to be long-term, not a job where you are worried about being laid off and knowing you most likely won’t get another position.”
 - n “IT is treated like ‘maintenance’ work by most companies. Do YOU want to spend your life struggling to keep a job as a compu-janitor?”
-

October 27, 2005

ICER

And one more blog...

n “It’s thankless at work, it’s thankless at home. No one gets it but your fellow IT people and you don’t see them because you’re at the desk all day long. Run for the damn hills. I am VERY good at this job. But if I known then what I know now, I would have been a jock, or an artist, or a hermit. ANYTHING but an IT guy.”

October 27, 2005

ICER

Need for CS degrees?

OPINION



VIRGINIA
ROBBINS

“I completely agree that today's businesses require highly skilled, experienced professionals. I remain unconvinced that hiring computer science degree holders is the best way to achieve that. At my firm, roughly half of my technical staff graduated with English or other liberal arts degrees, and the other half have computer science, mathematics, MIS or other technical degrees. A smaller number have both, with either an undergraduate liberal arts degree and a technical master's degree or vice versa. I've found no correlation between degree and competency.”

ComputerWorld, SEPTEMBER 26, 2005

October 27, 2005

ICER

H-1B Visas

Last week the U.S. Senate Judiciary Committee approved legislation that would expand the cap on H-1B skilled-worker visas from 65,000 to 95,000 in fiscal year 2006.

IEEE-USA is against this proposal. IEEE-USA has said the H-1B program takes jobs away from U.S. workers. U.S. IT and electrotechnology professionals saw a 1.5% decrease in their salaries in 2003, the first decrease since IEEE-USA began surveying members in 1972, the group said in December. IEEE-USA blamed H-1B visas, outsourcing and other factors for the salary decrease.

October 27, 2005

ICER

Money used to be the good news...

“ For the fourth year in a row, IT workers across the board received only modest raises – their pay increased by an average of just 3% in 2005, matching last year’s average salary increase.... IT raises still lagged slightly behind the average of about 3.2% for all U.S. workers as reported by the Bureau of Labor Statistics. While the majority of respondents (69%) said their 2004 base salary increased from one year ago, 31% experienced either no change in salary or had their pay cut.”

ComputerWorld’s 19th Annual Salary Survey, October 24, 2005

October 27, 2005

ICER

Technology savvy kids?

“Even if young people don't know that salaries and job openings in computer science are on the rise, they're hooked on so much technology -- cell phones, digital music players, instant messaging, Internet browsing -- that it's puzzling why more don't want to grow up to be programmers.”

Bill Gates: Microsoft Research Faculty Summit

October 27, 2005

ICER

Who are these young people?



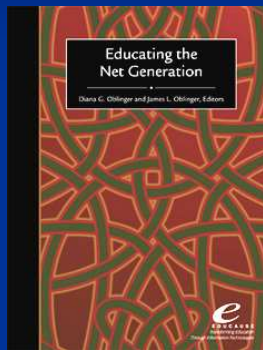
Diana Oblinger

- n Vice President for EDUCAUSE
- n Adjunct Professor of Adult and Community College Education at North Carolina State University
- n Vice President for Information Resources and the Chief Information Officer for the 16-campus University of North Carolina
- n Executive Director of Higher Education for Microsoft
- n IBM Director of the Institute for Academic Technology

October 27, 2005

ICER

Educating the Net Generation



Is it Age or It: First Steps Toward Understanding the Net Generation

Technology and Learning Expectations of the Net Generation

Using Technology as a Learning Tool, Not Just the Cool New Thing

The Student's Perspective

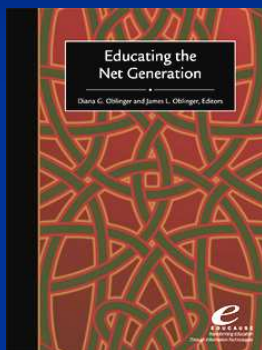
Preparing the Academy of Today for the Learner of Tomorrow

Convenience, Communications, and Control: How Students Use Technology

October 27, 2005

ICER

Educating the Net Generation



The Real Versus the Possible: Closing the Gaps in Engagement and Learning

Curricula Designed to Meet 21st Century Expectations

Support Services for the Net Generation

Faculty Development for the Net Generation

Learning Spaces

Net Generation Students and Libraries

The New Academy

Planning for Neomillennial Learning Styles: Implications for Investments in Technology and Faculty

October 27, 2005

ICER



The Net Generation



The Millennials	Our Generation
Multitasking	One thing at a time
Pictures, sound, video	Text
Action	Reflection
Social	Individual
Speed	Deliberation
Peer to peer	Peer review

October 27, 2005

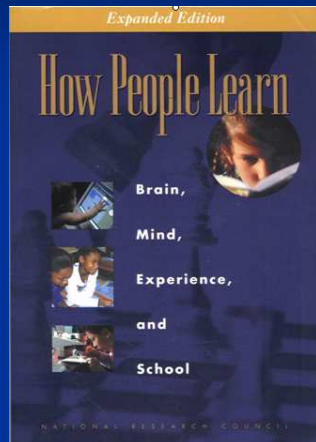
ICER



October 27, 2005

ICER

Constructivist Theory



Key findings:

“Students come to the classroom with preconceptions about how the world works. If their initial understanding is not engaged, they may fail to grasp the new concepts and information that are taught, or they may learn them for purposes of a test but revert to their preconceptions outside the classroom.”

October 27, 2005

ICER

Constructivist Theory

Implies that learning is best when it is:

- n **Contextual:** taking into account the student's understanding
- n **Active:** engaging students in learning activities that use analysis, debate, and criticism (as opposed to simply memorization) to receive and test information
- n **Social:** Using discussion, direct interaction with experts and peers, and team-based projects

October 27, 2005

ICER

How did we learn?



October 27, 2005

ICER

How do we teach?

- n The same way we were taught?
- n The way we prefer to learn?



Visual Learners
Auditory Learners
Kinesthetic Learners

October 27, 2005

ICER

How do children learn today?



October 27, 2005

ICER

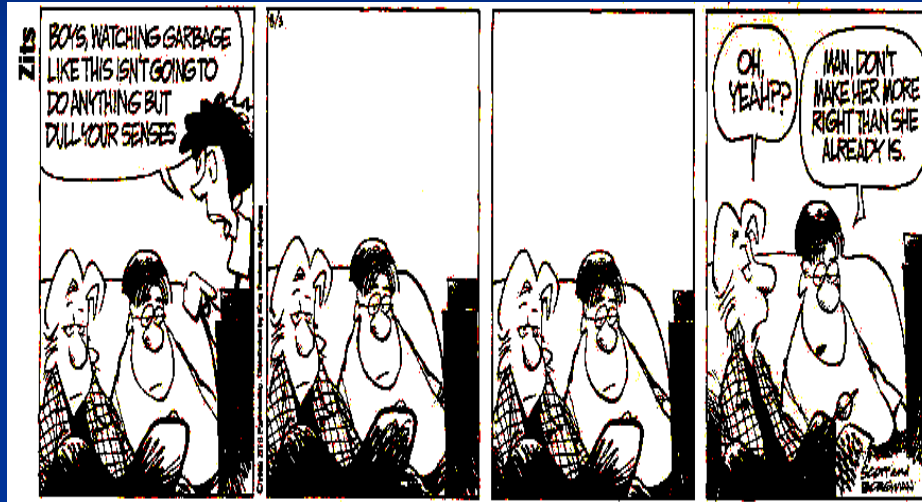
How do our students learn?



October 27, 2005

ICER

The Impact of Television on our Current Students



Remember our videos?



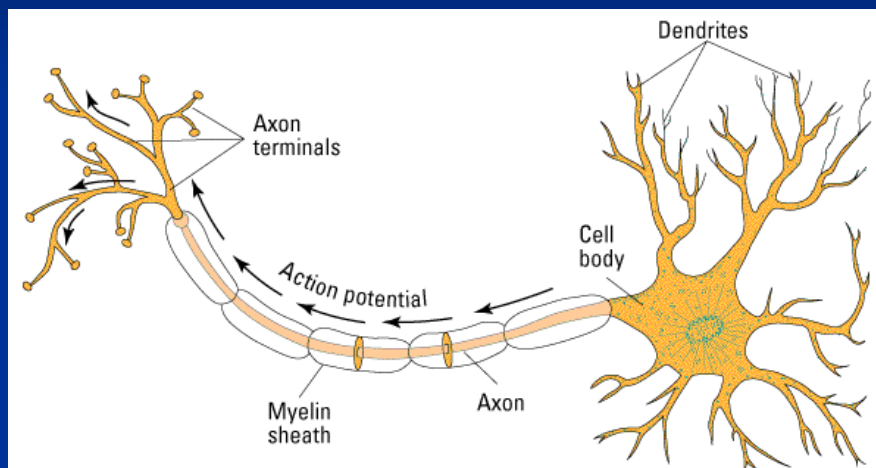
What are they watching?



October 27, 2005

ICER

Back to the Basics The Structure of a Neuron



October 27, 2005

ICER

The Brain

Regions of the Brain

Limbic system

- Thalamus
- Hypothalamus
- Pituitary
- Amygdala
- Hippocampus
- Cerebellum

Brain stem

- Pons
- Reticular formation
- Medula

Cerebral cortex

Spinal cord

October 27, 2005

ICER

Regions of the Brain

Association cortex

Motor cortex

Somatosensory cortex

Association cortex

Frontal lobe

Parietal lobe

Broca's area

Temporal lobe

Occipital lobe

Auditory cortex

Wernicke's area

Visual cortex

October 27, 2005

ICER

And time for a commercial




October 27, 2005

ICER

So we should...

Keep the frontal lobe stimulated!

- ü Visuals, videos
 - ü Changing stimulus
 - ü Group work
 - ü Collaborative work
 - ü Experiential work
 - ü Role playing
 - ü Competition and games
 - ü Evaluation, critiquing
- 
- ü Brainstorming
 - ü Blogs
 - ü Case studies
 - ü Online discussion
 - ü Adaptive courses

October 27, 2005

ICER

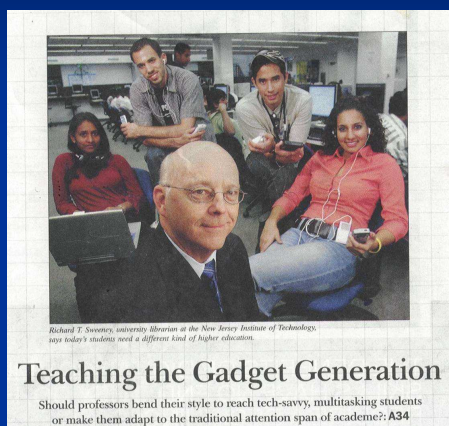
“Microburst” Teaching and Learning

Combine various teaching styles and methods to interest and motivate students with different and sometimes disparate learning styles for the ultimate purpose of enhancing and strengthening the learning process

October 27, 2005

ICER

But who should change?



Chronicle of Higher Education, October 7, 2005

October 27, 2005

ICER

Net Gen Attitudes

- n "My laptop follows me wherever I go. The world should be wireless."
- n "If I can't Google it, it probably isn't worth knowing."
- n "Instead of focusing on required attendance, teachers should focus on providing students with the tools to make them successful and should let them be responsible for their own learning."

October 27, 2005

ICER

More Net Gen Attitudes

- n "I would like to have the libraries be completely Web-based, with books that have highlightable text."
- n "I carry the Internet in my pocket, and boy, do I need it."
- n "Let me know what it is that's required of me, and then let me get there. Facilitate my way there, provide me with the resources, be available—on the phone or online. It would be great if we could IM our professors. "

October 27, 2005

ICER

And one more...

- n “In teaching, there seems to be a lot of emphasis on learning facts, with the teacher communicating facts to the students. But *Wikipedia* has pretty much all the facts a student needs. Or a Google search can provide the facts. One way or another, the Internet has the facts. What I’d love to see more of in the classroom is experience. If my professor has been a physicist for twenty years, has been solving calculus problems, I’d like the professor to share with me some of these things—share with me the approaches taken, the thought processes involved. That would definitely help students be able to work with the materials to solve the assigned problems.”

October 27, 2005

ICER

But lots of issues

- n Students come into CS to do gaming
- n Students are working lots of hours
- n Students come to campus with lots of baggage
- n Have to teach the basics and all the new stuff
- n Where do online courses come into the mix?

October 27, 2005

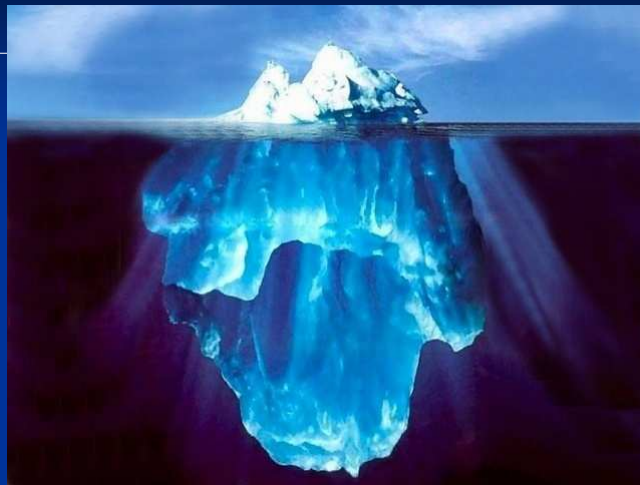
ICER

What do we do?

- n Make science and math interesting at a young age
- n Need industry to step forward
- n Educate faculty on how to reach out to these students – regional workshops – best practices
- n Change our image to make it more appealing – connecting to other disciplines and real-world applications
- n Fund innovations at the local level

October 27, 2005

ICER



October 27, 2005

ICER

Computing: A 21st Century Liberal Arts Education

Dan Reed
Dan_Reed@unc.edu

Chancellor's Eminent Professor
Vice Chancellor for IT
University of North Carolina at Chapel Hill

Director, Renaissance Computing Institute (RENCI)
Duke University
North Carolina State University
University of North Carolina at Chapel Hill

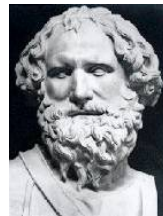
Chair, Board of Directors
Computing Research Association (CRA)



Information Technology: The Lever

- **The 21st century is about knowledge economies**
 - managing information for competitive advantage
- **Universities are in the knowledge business!**
 - creation, preservation, transmission, and application
- **IT is about knowledge management and creation**
 - education, research, service and business processes
- **Globalization is a challenge and opportunity**
 - we made it possible; we should capitalize on it

“Give me a place to stand and a lever
long enough, and I will move the world.”
Archimedes, 287-212 BC



Why Are We Here?



I am often asked, “What made you become scientist?” But I can't stand far enough away from myself to give a really satisfactory answer, for I cannot distinctly remember a time when I did not think that a scientist was the most exciting possible thing to be.

Sir Peter Medawar



The Story Does Matter

Sashimi



Dead Tuna



Exemplar 21st Century Challenges

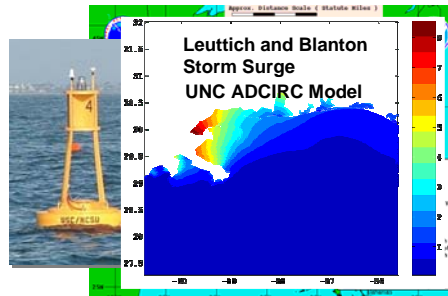
- **Population growth**
 - severe weather sensitivity
 - statewide impact
 - geobiology and environment
 - economics and finance
 - sociology and policy

- **Economics and health care**
 - longitudinal public health data
 - environmental interactions
 - genetic susceptibility
 - heart disease, cancer, Alzheimer's
 - privacy and insurance
 - public policy and coordination



Severe Storm Modeling

- **\$10T U.S. economy**
 - 40% is adversely affected by weather and climate
- **\$1M in loss to evacuate each mile of coastline**
 - we now over warn by 3X!
 - average over warning is 200 miles (\$200M/event)
- **Multiple models**
 - atmosphere, ocean, geography
 - biology, fishing, environment
 - economic and social
- **Goal**
 - timely and accurate forecasts
 - using dynamic adaptation
- **Attributes**
 - integrated monitoring and analysis
 - data capture and adaptive analysis
- **LEAD Grid**



- Linked Environments for Atmospheric Discovery
- Oklahoma, Indiana, UCAR, Colorado State, Howard, Alabama
 - Millersville, NCSA, North Carolina



Surrounded by Invisibility

“Eventually, living in a world of continuous computing will be like wearing eyeglasses: the rims are always visible, but the wearer forgets she has them on—even though they’re the only things making the world clear.”

*Wade Roush, MIT
Technology Review*



Understanding the Future

- **Some rules of thumb**
 - in the *near term*, we *overestimate* change
 - in the *long term*, we *underestimate* changes
- **Outside their field of expertise**
 - experts are often better at predictions
 - the contra-Delphi effect
- ***Inventing the future* is far more successful**
 - recognize exponentials
 - quantitative change brings qualitative change
- **Technological and social change**
 - move at different rates and have differing consequences
- **Consider digital music**
 - born of storage technology advances
 - bred social and business change



Digital Reality: The Exponentials



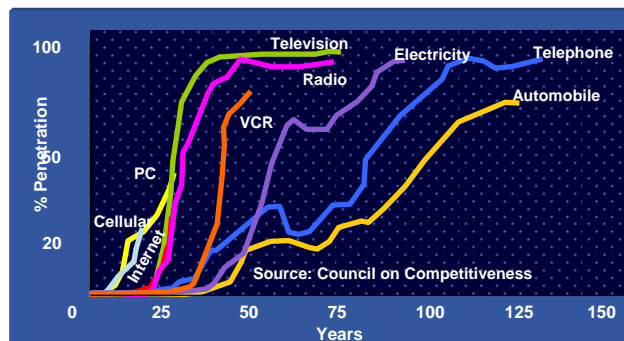
- **Megabyte**
 - a small novel
- **Gigabyte**
 - a pickup truck filled with paper or a DVD
- **Terabyte: one thousand gigabytes – ~\$1000 today**
 - the text in one million books
 - entire U.S. Library of Congress is ~ten terabytes of text
- **Petabyte: one thousand terabytes**
 - 1-2 petabytes equals all academic research library holdings
 - coming soon to a pocket near you!
 - *soon routinely generated annually by many scientific instruments*
- **Exabyte: one thousand petabytes**
 - 5 exabytes of words spoken in the history of humanity
- See www.sims.berkeley.edu/research/projects/how-much-info-2003/



Source: Hal Varian, UC-Berkeley



The Pace of Innovation is Quickening



- **Social compacts and acceptable use**
 - historically, generally over a generation
 - generational internalization is no longer possible
- **Concomitant economic dislocation**
 - learning for a lifetime

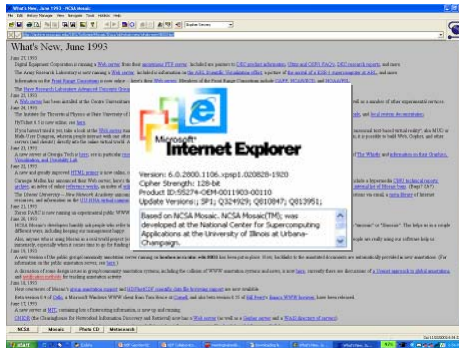


NCSA Mosaic: Almost Twelve Years

The New York Times

•December 8, 1993, C Section (Front Page)

–John Markoff, “A Free and Simple Computer Link - NCSA's Mosaic Program”



Information and Social Processes

- **Google**
 - it's a search engine, it's a verb, ...
- **Blogs**
 - published self-expression
- **Instant Messenger**
 - social networks
- **Wireless messaging**
 - semi-synchronous
- **Internet commerce**
 - the dot.com boom/bust
 - EBay, Amazon
- **Spam, phishing, ...**
 - anti-social behavior



the official **Kerry-Edwards** blog



Economic Dislocation: What and Why?

- **The world shrank**
 - ship, railroad, airplane
 - telecommunications
- **Information “friction” declined**
 - international interactions at local cost
 - a long distance call once was a big deal
 - 24x7 asset use by multinationals
- **Web of interdependencies increased**
 - NAFTA, Kyoto accord, oil reserves
 - commercial products (“Made in USA”)
- **Rate of change accelerated**
 - globalization and competitive pressures
 - few “life long” jobs remain
 - rapid, come as you are response
- **Implications**

U.S. economy and universities?



The New York Times

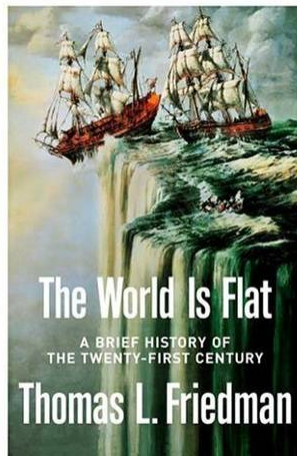
November 2, 2004

Elizabeth Becker, “Textile Quotas to End, Punishing Carolina Towns”

“Leann Harrington, a former textile worker and now a waitress in Kannapolis, N.C., said that ‘the middle class is pretty much gone here.’”



The World Is Flat

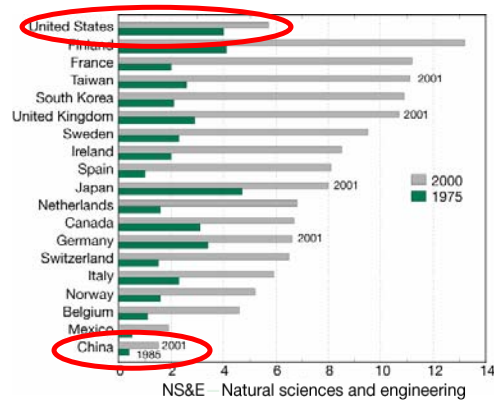


- **U.S. competition is not among**
 - Boston, Atlanta, Chicago, Los Angeles, ...
- **The competition is *global***
 - Bangalore, Mumbai, Shanghai, Seoul, ...
- **Costs of information flow**
 - are approaching zero
- **Global economic winners will**
 - have a better trained workforce
 - have the ability to participate globally
 - be horizontally, not vertically integrated
 - nobody is best at everything



Science and Engineering Degrees

- **Population ratios**
 - 24 year olds
 - NS&E degrees
 - Natural Science & Engineering
- **Changing behavior**
 - U.S. implications
 - globalization and innovation
 - 21st century economy



Source: NSF S&E Indicators, 2004



Evolving University Roles

- **American university “eras”**
 - pre and post-colonial
 - private and original state universities
 - land grant
 - many state universities
 - post World War II
 - GI bill and educational “democratization”
 - today, the fourth wave
 - economic drivers and continual re-education
- **A new compact with the citizens**
 - *lifelong education and economic competitiveness*
 - *knowledge economy leverage*
 - *value chain enhancement*
- **Intelligent application of IT is a powerful force**



Roles of Great Universities

- **Frame and lead the debate on critical issues**
 - shape state, national and international positions
- **Train tomorrow's leaders**
 - public and private
- **Enrich the human experience**
 - scholarly, cultural, social and recreational activities
- **Nurture and broaden participation**
 - the common wellspring of humanity
- **Produce and transfer new knowledge**
 - the raw material of the knowledge economy
- **Sustain lifelong education**
 - knowledge for a lifetime and refreshed skills for a profession
- **Catalyze economic development**
 - job creation and corporate competitiveness



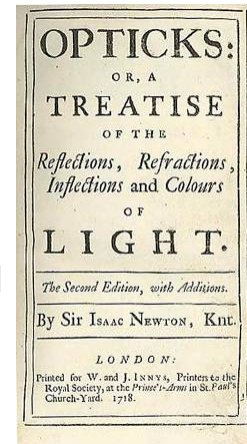
What of Computer Science?

- **What are we called?**
 - computer science
 - computer engineering
 - computing
 - information technology
 - information science
 - informatics
- **What are we?**
 - science
 - engineering
 - technology
 - vocational
 - theoretical
 - experimental
 - chimera
- **What is our core?**
 - theory, software
 - architecture, hardware
 - graphics, AI, HCI, NA
 - computational science
- **What is our vision?**
 - engagement
 - inclusion
 - excitement
- **What is our future?**
 - grand challenges
 - collaborations
 - redefinition

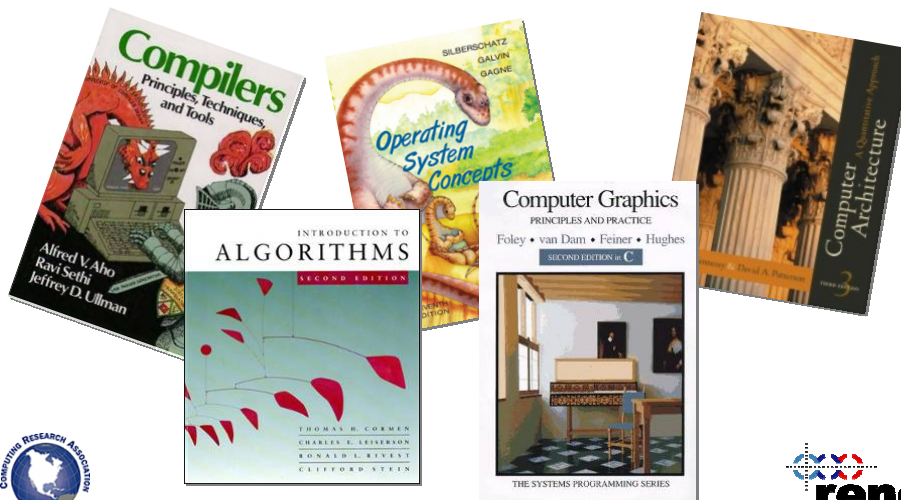


The Classical Syllabus

- **Classical education**
 - *trivium*
 - grammar, logic and rhetoric
 - *quadrivium*
 - arithmetic, geometry, music and astronomy
 - and fluency in Latin and Greek
- **Natural philosophy then appeared**
 - Robert Boyle
 - Francis Bacon
 - Isaac Newton



The Standard CS Syllabus

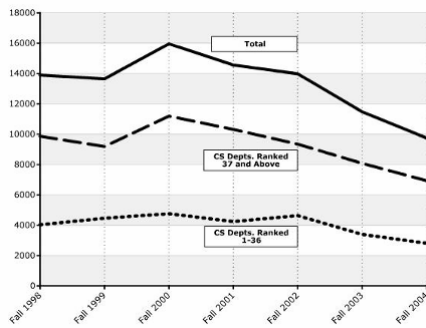


The Infinite Onion

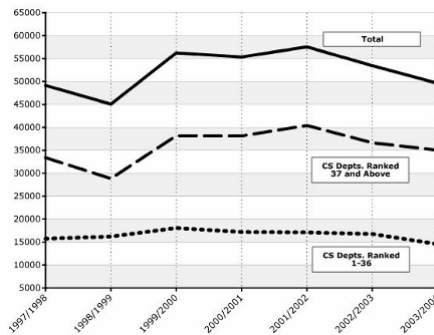
- **We always add ...**
 - new courses
 - new technologies
 - new paradigms
 - new layers
- **Do we ever delete?**
 - core plane design
 - code overlays
 - FORTRAN (in some cases)
- **What are the CS “Latin and Greek?”**
 - time to rethink and move forward
- **CS/IT: the Renaissance skill of the 21st century**
 - problem solving is the “liberal arts” skill
 - universal applicability and domain connections



Enrollment Trends (CRA Data)



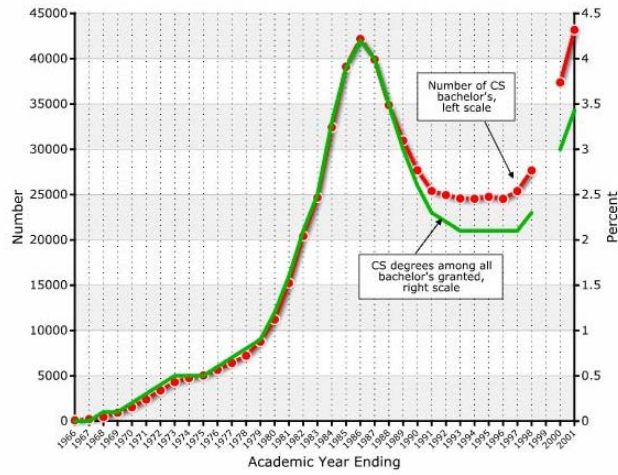
Newly Declared CS Majors



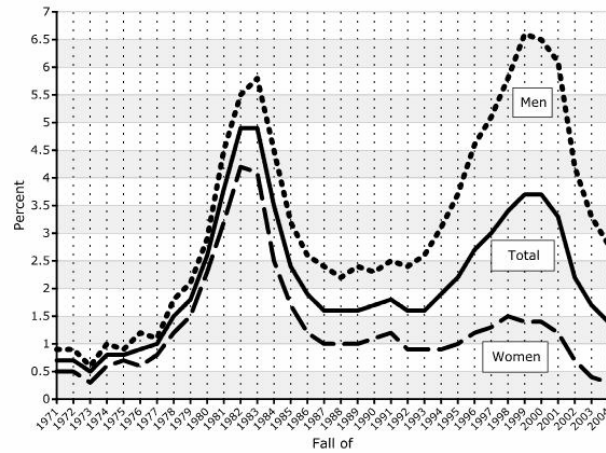
Undergraduate CS Enrollments



Degrees Awarded (NSF Data)



Probable Majors By First Year Students (HERI/UCLA Data)



Global Challenges

- **Rising Above The Gathering Storm: Energizing and Employing America for a Brighter Economic Future**

- “What are the top ten actions, in priority order, that federal policy makers could take to enhance the science and technology enterprise so the United States can successfully compete, prosper and be secure in the global community of the 21st Century? What implementation strategy, with several concrete steps, could be used to implement each of those actions?”



books.nap.edu/catalog/11463.html



PITAC Report Contents

- **Computational Science: Ensuring America's Competitiveness**
 1. *A Wake-up Call: The Challenges to U.S. Preeminence and Competitiveness*
 2. *Medieval or Modern? Research and Education Structures for the 21st Century*
 3. *Multi-decade Roadmap for Computational Science*
 4. *Sustained Infrastructure for Discovery and Competitiveness*
 5. *Research and Development Challenges*
- **Two key appendices**
 - *Examples of Computational Science at Work*
 - *Computational Science Warnings – A Message Rarely Heeded*

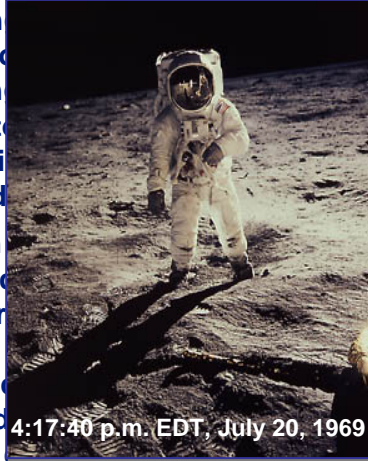


Available at www.nitrd.gov



Extraordinary Dreams

Between them, the five engines were now vaporizing the condensation. ... From the mystifying seemed to tower of iron no sound. And then and thud spectators their feet watched ascended the planet.



Ascent to the Moon



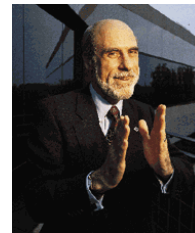
ARPANET



Len Kleinrock



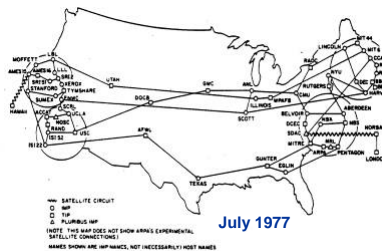
BBN IMP Team



Vint Cerf



Larry Roberts



Note the timescale!



Bob Kahn



DARPA Grand Challenge Race

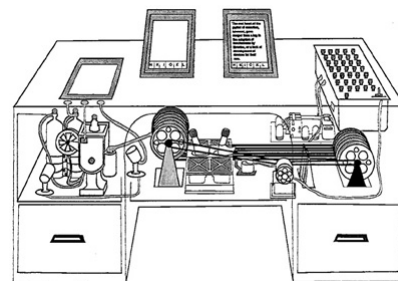


Memex: Still Prescient

“Consider a future device for individual use, which is a sort of mechanized private file and library. It needs a name, and to coin one at random, “memex” will do. A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.”

Vannevar Bush

“As We May Think,” 1945



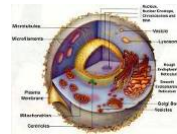
The Importance of Dreams

- **Defining characteristics of humanity?**
 - opposable thumbs? Perhaps
 - tool builders? Perhaps
 - self-awareness? Closer, but not quite
- **The ability to dream what *could be***
 - The Library at Alexandria
 - capturing all human knowledge
 - *Principia Mathematica* (Whitehead and Russell)
 - led to Gödel and the limits of logical systems
 - Apollo program, even given Cold War politics
 - Kennedy's advisors grasped the essence of the vision
 - The Central Dogma of biology
 - human genetic sequencing and engineering
 - The Standard Model of physics
 - dark matter, dark energy and the Theory of Everything



Some Informatics Grand Challenges

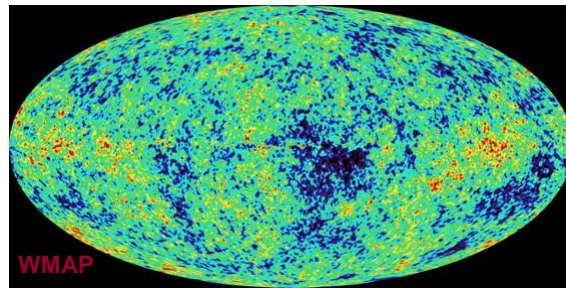
1. **Ubiquitous invisibility**
 - successful technologies become “invisible”
 - composable, interoperable systems
 2. **Intelligence amplification (Memex)**
 - the right information at the right time
 - seamless modality transduction, situated and mobile
 3. **Predictive *in silico* biological models**
 - the “other” artificial life
 - multidisciplinary modeling and integration
 4. **The Universe in a Box**
 - origins and alternatives
 - the theory of everything (TOE)
 5. **The Cultural Encyclopedia**
 - cultural history, context and the digital village
 6. **Semi-automated extra-solar exploration**
 - the Fermi Paradox and universal constructors
- Grand AI, our long-term fascination**
deep questions about thinking



Ask The Big Questions

Our immediate neighborhood we know intimately. But with increasing distance our knowledge fades. ...The search will continue. The urge is older than history. It is not satisfied, and it will not be denied.

Edwin Hubble



Southeast Region Workshop on "Integrative
Computing Education and Research (ICER):
Preparing IT Graduates for 2010 and Beyond,"



Murali Varanasi
Oct 28, 2005



Learning

- n Years ago, most theories about teaching and learning were that if facts were repeated enough, then students would memorize them, and this was learning.
- n Since the 1970's the field of cognitive psychology has taken a different approach -- looking at what people believe about what they are studying, how they go about solving problems, and how aware they are of whether they understand what they are reading.



Learning - II

- n Problem-solving skills in one subject are different from those in other subjects.
- n Learning lasts when the student understands the material, not just memorizes it. Information needs to be presented in small chunks so that working memory can process it.
- n Students need immediate practice to move information from working memory to long-term memory.
- n Background knowledge is vital – it affects memory, reading, thinking, and problem solving. People have informal beliefs about how the world works, which interferes with learning.



Three Findings

- n Students come to the classroom with preconceptions about how the world works. Build on this!
- n Develop student competence with: (a) a deep foundation of factual knowledge, (b) facts and ideas in context, and (c) organization of knowledge to enable retrieval and application.
- n A "metacognitive" approach to instruction can help students learn better.



Implications

- n Faculty must exploit and build upon students' prior understanding.
- n Faculty must treat subject matter in depth with examples.
- n The teaching of metacognitive skills should be integrated throughout the curriculum.



Your Path for Most Effective Learning is Through Knowing

- n Yourself
- n Your capacity to learn
- n The process you have successfully used in the past
- n Your interest in, and knowledge of, the subject you wish to learn



Begin With the Past

- n Past learning experiences:
 - n Like to read? Solve problems? Memorize? Recite? Interpret? Speak to groups?
 - n Know how to summarize?
 - n Ask questions about what you studied?
 - n Review?
 - n Have access to information from a variety of sources?
 - n Like quiet or study groups?
 - n Need several brief study sessions, or one longer one?
- n What are your study habits? How did they evolve? Which worked best? Worst?
- n How did you communicate what you learned best? Through a written test, a term paper, an interview?



Proceed to the Present

- n How interested are you in this?
- n How much time do you want to spend learning this?
- n What competes for your attention?
- n Are the circumstances right for success?
- n What can you control, and what is outside your control?
- n Can you change these conditions for success?
- n What affects your dedication to learning this?
- n Do you have a plan? Does your plan consider your past experience and learning style?



Learning Styles

- n Visual - graphic
- n Auditory - hearing
- n Kinesthetic - movement



Know How You think

- n Knowing how you think is important, especially for engineers.
- n Know background behind your reasoning.
- n As the reasoning is self dependent, take out some time to analyze your skills.
 - n Know what you are doing?
 - n Strategies employed
 - n The approach used
 - n The backup selected
 - n The emotions involved in them
 - n Confidence level improving after a successive attempt
 - n Confidence lost after a failure.



Thinking Styles

- n Know that styles are preferences to use abilities, but not instincts.
- n A match between styles and abilities creates synergy, which is more than the sum of parts.
- n Styles vary across tasks and situations
- n Styles are socialized
- n Styles vary across the life span
- n Styles are teachable
- n Some styles may better suit you but do not fit the learning environment



Forms of Thinking

MONARCHIC STYLE:

- n They get motivated by the thinking style
- n They are single minded.
- n They are too decisive
- n They are people who have others coming for advice.

HIERARCHIC STYLE:

- n These people get motivated by hierarchy of goals
- n They are systematic and organized people.



Memory

- n Memory is used for retention, storage, and retrieval of information.
- n Knowledge is different from memory. It refers to information that has been successfully retained in storage and later remembered.
- n There are quite a few theories regarding memory.



Memory

- n Mnemonics
- n Learn the pattern of memory
- n Learn how you have improved your memorization skills



Short Term Memory

- n This is the working memory.
- n Working memory is limited in space and stores information for probably 12 hours.
- n Information needs revision to hold in the long term memory.
- n Sleep refreshes the short term memory.



Long Term Memory

- n The long term memory is limitless.
- n The memory is organized in the following manner
 - n Episodic memory
 - n Semantic memory

Short and long term memory are highly integrated. There are links to the short term memory and the long-term memory.



...Continued

Episodic Memory

- n Autobiographical in nature.
- n Record of your personal experiences.
- n Childhood memories

Semantic Memory

- n Organizing rules.
- n Practicing habits



Definition of Creativity

- n The ability to create. (dictionary definition)
- n Influencing Static Knowledge by using Dynamic Intelligence.
- n Creative Thinking
 - n Improvising information
 - n Synthesizing something, which is not present by logical conclusions, deriving it from elements already present.
 - n Anticipating by logical conclusions.



Creativity - II


- n Creativity is a trait we intuitively associate with intelligence.
- n Creativity is easy to observe.
- n It is the ability to synthesize our existing knowledge in such a way as to create something new.
- n Perceiving a lack of creativity stifles discovery



Creative Thinking Abilities


<http://www.ldrc.ca/projects/atutor/?cid=257#define>

- n **Fluency:** The ability to produce many responses to an open-ended question or problem.
- n **Flexibility:** The ability to generate unconventional ideas, and to view a situation from different perspectives.
- n **Originality:** The ability to produce unique, unusual, or novel responses.
- n **Elaboration:** The ability to add rich and elaborate detail to an idea.
- n **Visualization:** The ability to imagine and manipulate images and ideas from different perspectives.




Creative Processes

- n Transformation: The ability to change one thing or idea into another, to see new meanings, applications, and implications of something already in place.
- n Intuition: The ability to see relationships or make connections based on partial information.
- n Synthesis: The ability to combine parts into a coherent whole.




Mnemonics

- n Associate thoughts- associate your thoughts and relate them to things you are familiar with
- n Concreteness of items associated- relate to something you know.
- n Automated – learn principles and use them mechanically



Problem Solving

- n How good are you at problem solving?
- n When you face a new problem, are you involved in it and do you solve it quickly?
- n Do you learn the formulae first and then apply?
- n Do you think strong concepts are important in solving problems?



Consider the Process

- n What is the heading or title?
- n What are key words that jump out?
- n Do I understand them?



The Subject Matter

- n What do you already know?
- n Are there related subjects?
- n What kinds of resources and information will help me?
- n As you study, do you check for understanding?
- n Should you go more quickly or slow down?
- n If you don't understand, do you ask why?

[more...](#)



The Subject Matter - II

- n Do you summarize?
- n Do you ask whether it is logical?
- n Do you stop and evaluate (agree/disagree)?
- n Do you need time to think it over and return later?
- n Do you need to discuss it with other "learners" in order to process the information?
- n Do you need to consult a subject-matter expert?



Build in Review

- n What did you do right?
- n What could you make better?
- n Did your plan coincide with how you utilize your strengths and weaknesses?
- n Did you choose the right conditions?
- n Did you follow through?
- n Did you succeed?
- n Did you celebrate your success?