Report of NSF Workshop on
Integrative Computing Education and Research
Northeast Workshop

http://gaia.cs.umass.edu/nsf_icer_ne

Any opinions, findings, conclusions or recommendations expressed in this Report are those of the authors and do not necessarily reflect the views of the authors’ institutions or the NSF.

The NSF Workshop on Integrative Computing Education and Research (Northeast) was supported by NSF under grants CNS-0548789 and CNS-0548575.
Authors and Contributors

The following individuals attended the NSF Workshop on Integrative Computing Education and Research and contributed to the writing of this report.

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rick Adrion</td>
<td>U. Massachusetts</td>
<td><a href="mailto:adrion@cs.umass.edu">adrion@cs.umass.edu</a></td>
</tr>
<tr>
<td>Bob Aiken</td>
<td>Temple University</td>
<td><a href="mailto:aiken@joda.cis.temple.edu">aiken@joda.cis.temple.edu</a></td>
</tr>
<tr>
<td>Andy Bernat</td>
<td>CRA</td>
<td><a href="mailto:abernat@cra.org">abernat@cra.org</a></td>
</tr>
<tr>
<td>Jeffrey C. Brown</td>
<td>Raytheon</td>
<td><a href="mailto:jeffrey_c_brown@raytheon.com">jeffrey_c_brown@raytheon.com</a></td>
</tr>
<tr>
<td>Steve Cooper</td>
<td>St. Josephs University</td>
<td><a href="mailto:scooper@sju.edu">scooper@sju.edu</a></td>
</tr>
<tr>
<td>Mike Dunn</td>
<td>Indiana University</td>
<td><a href="mailto:dunn@indiana.edu">dunn@indiana.edu</a></td>
</tr>
<tr>
<td>Mary Finlay</td>
<td>Partners Health Care</td>
<td><a href="mailto:mfinlay@partners.org">mfinlay@partners.org</a></td>
</tr>
<tr>
<td>Roscoe Giles</td>
<td>Boston University</td>
<td><a href="mailto:roscoc@bu.edu">roscoc@bu.edu</a></td>
</tr>
<tr>
<td>David Gries</td>
<td>Cornell University</td>
<td><a href="mailto:gries@cs.cornell.edu">gries@cs.cornell.edu</a></td>
</tr>
<tr>
<td>Charles Kelemen</td>
<td>Swarthmore</td>
<td><a href="mailto:cfk@cs.swarthmore.edu">cfk@cs.swarthmore.edu</a></td>
</tr>
<tr>
<td>Shriram Krishnamurthy</td>
<td>Brown University</td>
<td><a href="mailto:sk@cs.brown.edu">sk@cs.brown.edu</a></td>
</tr>
<tr>
<td>Deepak Kumar</td>
<td>Bryn Mar College</td>
<td><a href="mailto:dkumar@brynmawr.edu">dkumar@brynmawr.edu</a></td>
</tr>
<tr>
<td>Jim Kurose</td>
<td>U. Massachusetts</td>
<td><a href="mailto:kurose@cs.umass.edu">kurose@cs.umass.edu</a></td>
</tr>
<tr>
<td>Andrea Lawrence</td>
<td>Spelman College</td>
<td><a href="mailto:lawrence@spelman.edu">lawrence@spelman.edu</a></td>
</tr>
<tr>
<td>Louis Masi</td>
<td>IBM</td>
<td><a href="mailto:lmasi@us.ibm.com">lmasi@us.ibm.com</a></td>
</tr>
<tr>
<td>Dan McCracken</td>
<td>City University of NY</td>
<td><a href="mailto:mccracken@cs.ccnuy.cuny.edu">mccracken@cs.ccnuy.cuny.edu</a></td>
</tr>
<tr>
<td>Susan Merritt</td>
<td>Pace University</td>
<td><a href="mailto:smerritt@pace.edu">smerritt@pace.edu</a></td>
</tr>
<tr>
<td>Tom Murtaugh</td>
<td>Williams College</td>
<td><a href="mailto:tom@cs.williams.edu">tom@cs.williams.edu</a></td>
</tr>
<tr>
<td>Joyce Plotkin</td>
<td>Massachusetts Technology</td>
<td><a href="mailto:joyce@masstechleaders.org">joyce@masstechleaders.org</a></td>
</tr>
<tr>
<td>Jane Prey</td>
<td>Microsoft</td>
<td><a href="mailto:jprey@microsoft.com">jprey@microsoft.com</a></td>
</tr>
<tr>
<td>Barbara Ryder</td>
<td>Rutgers University</td>
<td><a href="mailto:ryder@cs.rutgers.edu">ryder@cs.rutgers.edu</a></td>
</tr>
<tr>
<td>Ra’ad Siraj</td>
<td>Eaton Vance Management</td>
<td><a href="mailto:rsiraj@eatonvance.com">rsiraj@eatonvance.com</a></td>
</tr>
<tr>
<td>Lynn Stein</td>
<td>Olin College of Engineering</td>
<td><a href="mailto:lynn.stein@olin.edu">lynn.stein@olin.edu</a></td>
</tr>
<tr>
<td>Lixin Tao</td>
<td>Pace University</td>
<td><a href="mailto:ltao@pace.edu">ltao@pace.edu</a></td>
</tr>
<tr>
<td>Virginia Teller</td>
<td>Hunter College</td>
<td><a href="mailto:vteller@hunter.cuny.edu">vteller@hunter.cuny.edu</a></td>
</tr>
<tr>
<td>James Thomas</td>
<td>Penn State</td>
<td><a href="mailto:jthomas@ist.psu">jthomas@ist.psu</a></td>
</tr>
<tr>
<td>Heikki Topi</td>
<td>Bentley College</td>
<td><a href="mailto:htopi@bentley.edu">htopi@bentley.edu</a></td>
</tr>
<tr>
<td>Klaus Sutner</td>
<td>CMU</td>
<td><a href="mailto:sutner@cs.cmu.edu">sutner@cs.cmu.edu</a></td>
</tr>
<tr>
<td>Mary Shaw</td>
<td>CMU</td>
<td><a href="mailto:mary.shaw@cs.cmu.edu">mary.shaw@cs.cmu.edu</a></td>
</tr>
<tr>
<td>Ursula Wolz</td>
<td>College of New Jersey</td>
<td><a href="mailto:woz@tcnj.edu">woz@tcnj.edu</a></td>
</tr>
</tbody>
</table>

Acknowledgments

We wish to express our appreciation to the NSF Directorate for Computer and Information Science and Engineering (CISE) and its Divisions of Computer and Networked Systems (CNS) for their support and encouragement. In particular, we wish to acknowledge the assistance and participation of the following individuals.
Thanks also to Laurie Connors and Alana Wiens of the University of Massachusetts for their help in the logistics of organizing and running this workshop.

The workshop organizers were Charles Kelemen (Swarthmore), Jim Kurose (University of Massachusetts, co-chair), Susan Merritt (Pace University), Joyce Plotkin (Massachusetts Technology Leadership Council), Barbara Ryder (Rutgers University), co-chair, Ursula Wolz, (The College of New Jersey)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>1</td>
</tr>
<tr>
<td>Authors and Contributors</td>
<td>2</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>2</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>4</td>
</tr>
<tr>
<td>Executive Summary</td>
<td>5</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>8</td>
</tr>
<tr>
<td>2. Looking Inward: The Traditional Computer Science Curriculum</td>
<td>10</td>
</tr>
<tr>
<td>2.1 The Introductory Course</td>
<td></td>
</tr>
<tr>
<td>2.2 The Bare Essentials and Enduring Fundamentals of Computer Science</td>
<td></td>
</tr>
<tr>
<td>2.3 Pedagogies and Learning Environments</td>
<td></td>
</tr>
<tr>
<td>2.4 Research versus Education: Promotion, Tenure, Funding</td>
<td></td>
</tr>
<tr>
<td>3. Looking Outward: The Computer Science Curriculum Broadly Interpreted</td>
<td>20</td>
</tr>
<tr>
<td>3.1 Challenges in Instantiating an Expansive Definition of Computer Science</td>
<td></td>
</tr>
<tr>
<td>3.2 Approaches to Cross Disciplinary Education and Teaching</td>
<td></td>
</tr>
<tr>
<td>3.3 Organizational Models for Computing in the Academy</td>
<td></td>
</tr>
<tr>
<td>4. Broadening Participation</td>
<td>27</td>
</tr>
<tr>
<td>4.1 The Pipeline Problem: K-12 Exposure to Computing</td>
<td></td>
</tr>
<tr>
<td>4.2 Appealing to Members of Under-represented Groups</td>
<td></td>
</tr>
<tr>
<td>4.3 Improving the Image of Computing</td>
<td></td>
</tr>
<tr>
<td>4.4 Relationship Between Industry and Academia</td>
<td></td>
</tr>
<tr>
<td>Appendix A: Workshop Agenda</td>
<td>40</td>
</tr>
</tbody>
</table>
Executive Summary

The discipline of Computer Science finds itself at a crossroads today. We face challenges in the form of declining enrollments in undergraduate Computer Science majors and in high school AP Computer Science classes, under-participation in Computer Science by large segments of our society, and a public that perceives our discipline as the province of Dilbert-like geeks whose jobs may be outsourced in the near future. The rapid pace of technological change makes it difficult to capture material in a curriculum that is not quickly outdated. There are differences in opinion about what constitutes the core of computing and how we can best produce intellectually-agile graduates who are prepared for life-long learning.

Yet it is also a time of tremendous opportunity as computing, in its myriad forms, infuses and transforms disciplines not only within science and engineering but in numerous other academic disciplines as well. Increasingly, computing ubiquitously permeates our work and our culture. Indeed, many of the opportunities identified in this report result from an expansive definition of Computer Science, including the application of Computer Science concepts and methodologies, not only in areas of science and engineering but in numerous other academic disciplines and beyond. This broad definition of Computer Science – one that we believe needs to be embraced by the community - includes computing disciplines (e.g., computer engineering, information systems, information technology and software engineering) that have often developed model curricula separately from Computer Science.

Against this backdrop of challenge and change, the NSF Northeast Workshop on Integrative Education and Research (ICER) was held on November 3–4, 2005 in Boston Massachusetts, bringing together 30 key stakeholders in CISE education (generally from the northeast region of the United States) to identify and discuss needed transformations in undergraduate computing education in the USA. These stakeholders included computing faculty, academic administrators, representatives from industry and professional computing societies, and funding organizations. While many of the academic participants were from Computer Science departments, a number of participants were from schools of Information Technology or other non-CS departments.

This report discusses the findings and recommendations of the workshop. We summarize these recommendations below. Background and amplifying discussion surrounding these recommendations can be found in sections 2, 3, and 4 of this report, where the recommendations are broken out in text boxes. Section 2 takes an introspective, inward-looking view at the traditional (rather narrowly-defined) Computer Science curriculum. Section 3 takes the more expansive outward-looking view of Computer Science to include the many forms of computing noted above. Section 4 discusses the challenging issue of broadening participation.

The first set of recommendations result from the introspective, inward-looking view of Computer Science. The topics covered include the introductory Computer Science course(s), the need to identify and teach the bare essentials and enduring fundamentals of Computer Science, computing pedagogy, including experiential and contextual learning, and the role of teaching in the academic reward system in Computer Science departments:
• **Recommendation 1**: Most first year courses for computing majors and non-majors are in need of radical redesign; the implications of such a redesign may ripple well beyond these introductory courses. At a national level, funding is needed for curricular experiments whose results and artifacts can be leveraged across institutions. Strong departmental support for faculty engaging in this activity (both as curriculum designers and as instructors) must be provided.

• **Recommendation 2**: Faced with an ever-increasing body of Computer Science knowledge, an increased need for project-based and interdisciplinary courses for our majors, and an increasing number of students who can take only a few Computer Science courses, we must recognize that it is not possible for us to teach everything a computing professional might need to know. *We cannot keep adding material to our curricula but instead must strip our discipline down to its bare essentials – the core skills, concepts, and methods that underlie our field.*

• **Recommendation 3**: There is a need to fund the experimentation and development of new pedagogies, research that evaluates the effectiveness of such new pedagogies, and the dissemination of the results of such efforts to the community.

• **Recommendation 4**: Innovative teaching and curriculum development must be valued as significant faculty contributions and achievements, and rewarded as such by P&T committees. Just as universities must be called on to emphasize the importance of educational activities and reflect this emphasis in their reward structure, so too must funding agencies. Needed education innovations can occur only when both universities and federal funding agencies are fully on board.

The second set of recommendations result from an outward-looking expansive view of Computer Science discussed in Section 3, a view that embraces many forms of computing as part of the Computer Science discipline, a view we believe needs to be adopted more broadly:

• **Recommendation 5**: Computer Science should take an expansive view of its domain. As a discipline, Computer Science should be considered both broad and fundamental, with application not only in the traditional science and engineering disciplines but in the arts, humanities, and social sciences. An expansive view of Computer Science fits well with the vision of a knowledge- and computing-intensive 21st century workplace.

• **Recommendation 6**: It is important for us to develop Computer Science courses that incorporate significant content from other disciplines and to create computing-centered modules that other disciplines can incorporate into their curricula. Agencies can provide funding for initial innovation, dissemination, and evaluation. We recommend that NSF or CRA convene a workshop to evaluate existing and proposed approaches, to collect best practices, and to develop guidelines for implementation and/or recommendations.

• **Recommendation 7**: A “one-size” fits all model of computing education is not sustainable in the long term. The curriculum must allow for a “multiple entry / multiple opportunity” approach that allows many different paths through an expansive computing curriculum. We must be able to accommodate students with diverse backgrounds (multiple entry points), diverse interests (multiple pathways through our major) and diverse educational goals (multiple exit points from our major).
• **Recommendation 8:** Several organizational models are evolving that bring a variety of computing- and information-centered disciplines as well as application areas under a single organizational unit, similar to engineering and business schools. We recommend that an agency such as NSF or the CRA and its Information Technology Deans Group undertake an effort to lead a community discussion of these various approaches in forms of a careful study of alternative organizational models for schools that bring together computing- and information-centered disciplines under one “roof”, as well as a meeting of heads of such departments in arts and sciences, engineering, and business schools (and their deans) to discuss content, organizational and structural issues.

The final section of this report, on *broadening participation*, explores several dimensions of this particularly pressing challenge. We discuss the inter-related issues of recruitment and retention of students from underrepresented groups, strengthening the K-12 pipeline, combatting the image crisis facing Computer Science, and the ways in which the academia and industry can interact to jointly address these challenges. Our recommendations are:

• **Recommendation 9:** Better resources for K-12 teacher training in computing must be made available. Standards for K-12 computing education must be enacted as a national educational priority, facilitated by a meeting sponsored by NSF and the National Academies. Educators, representatives from business, and (most importantly) state education commissioners would be key participants in such a workshop.

• **Recommendation 10:** Efforts must be expanded to recruit and retain women and under-represented minorities in computing disciplines. Needed are new introductory approaches to the discipline, the use of student-engaging teaching methods, emphasis on human-centric computing, and establishment of support structures on campus for members of these groups.

• **Recommendation 11:** A national “grand challenge” is needed to increase the percentage of under-represented groups in computing, focused at the high school level. A 5-year 50x50x5 plan will have 50 academic departments hold a 3-week course for high school women; 50 academic departments hold a 3-week course for high school minority students; and 50 academic departments hold a 2-week course for high school teachers.

• **Recommendation 12:** Improving the image of computing is key to attracting more students to the discipline and must be made a top priority within the Computer Science community. The CRA-led effort on *Improving the Image of Computer Science* could form the core of such an effort, but the set of participants should be expanded to include Information Technology councils and organizations.

• **Recommendation 13:** There are many potential opportunities for academia and industry to grow a more productive interaction in the educational arena, including increased internship, co-op, and entrepreneurial programs; two-way faculty/industry-practitioner exchanges; and industry/academic collaborations and advocacy on state-level activities (particularly for public institutions of higher education). Federal agencies can encourage such interactions by providing incentive matching for innovative industry/academic collaborations, creating 3-way (industry, academia, federal) partnerships.
The workshop participants hope that these thirteen recommendations, and the content in this report, can provide valuable guidance to the NSF and other interested organizations that together can initiate and enable a broadly-based, sustained effort to catalyze a high-impact transformation of undergraduate computing education across the nation.
1. Introduction

The discipline of Computer Science finds itself at a crossroads today. We face challenges in the form of well-documented declining enrollments in undergraduate Computer Science majors [CRA 2005a], under-participation in Computer Science by large segments of our society [CRA 2005b], and a public that perceives our discipline as the province of Dilbert-like geeks whose jobs may be outsourced in the near future. Our field is characterized by continuous change, rendering it difficult to capture material in a curriculum that is not quickly outdated, a situation that will continue into the foreseeable future. There are differences in opinion about what constitutes the core of computing and how we can best produce intellectually-agile graduates who are prepared for life-long learning.

Yet it is also a time of tremendous opportunity as computing, in its myriad forms, infuses and transforms disciplines not only within science and engineering but in numerous other academic disciplines as well. Increasingly, computing ubiquitously permeates our work and our culture. Indeed, many of the opportunities identified in this report result from an expansive definition of Computer Science, including the application of Computer Science concepts and methodologies, not only in areas of science and engineering but in numerous other academic disciplines and beyond. This broad definition of Computer Science – one that we believe needs to be embraced by the community - includes computing disciplines (e.g., computer engineering, information systems, information technology and software engineering) that have often developed model curricula separately from Computer Science.

Against this backdrop of challenge and change, the NSF Northeast Workshop on Integrative Education and Research (ICER) was held on November 3 and 4, 2005 in Boston Massachusetts, bringing together 30 key stakeholders in CISE education (generally from the northeast region of the United States) to identify and discuss needed transformations in undergraduate computing education in the USA. These stakeholders included computing faculty, academic administrators, representatives of industry and professional computing societies, and funding organizations. This report discusses the findings and recommendations of the workshop. Given the many challenges, needed changes, and opportunities identified, it is time to initiate a broadly-based, sustained effort to catalyze a high-impact transformation of undergraduate computing education across the nation. The workshop participants hope that recommendations and findings in this report can provide valuable guidance to the NSF and other interested organizations that together can initiate, enable, and sustain such an effort.

The remainder of this report divides broadly into three sections, each examining a different set of challenges.

- Section 2 looks inward at the traditional Computer Science curriculum. We focus on the introductory course(s), the need to identify and teach the bare essentials and enduring fundamentals of Computer Science, computing pedagogy, including experiential and contextual learning, and the role of teaching in the academic reward system in Computer Science departments.
Section 3 takes a more expansive view of Computer Science — a view we believe needs to be embraced more broadly — to include the application of Computer Science concepts and methodologies not only in areas of science and engineering but in numerous other academic disciplines and beyond. The increasing number of Information Technology programs and their relationship to traditional Computer Science programs was of particular interest to workshop participants. (We refer to Information Technology as IT in the body of this report.)

Section 4, on broadening participation, explores several dimensions of this particularly pressing challenge. We discuss the inter-related issues of recruitment and retention of students from underrepresented groups, strengthening the K-12 pipeline, combatting the image crisis facing Computer Science, and the ways in which the academia and industry can interact to jointly address these challenges.

Throughout Sections 2, 3, and 4, we break out our recommendations and findings in boxes accompanied by surrounding amplifying text.
2. Looking Inward: The Traditional Computer Science Curriculum

Workshop participants were unanimous in their opinion that the traditional Computer Science curriculum must change to better attract and educate computer and information technology professionals. It is imperative (and possible) to accomplish this change in a manner that will attract and better educate non-specialists as well. Simply put, computing education must change in the following ways:

(i) We need to change our introductory courses for both majors and non-majors.
(ii) We need to reduce our core by identifying the bare essentials and enduring fundamentals underlying our discipline.
(iii) We need to change our pedagogies and teaching methods.
(iv) We need to create opportunities for integrated curricula, so that computing builds bridges to adjacent disciplines (rather than standing alone like a silo).

We address items (i), (ii), and (iii) in Sections 2.1, 2.2, and 2.3, respectively. Item (iv) is the topic of Section 3.

2.1 The Introductory Course

The dramatic decline in enrollments in Computer Science courses, serious concerns regarding the perceived image of computing disciplines by the public, and the lack of participation from women and minorities has led us to recognize that there are still fundamental flaws in introductory Computer Science curricula. Introductory courses are essential both in directly attracting students to pursue majors in computing disciplines and in shaping the image of Computer Science in the wider academic community. These courses must engage the interest of a broad cross-section of the population, recognizing that it is neither necessary nor desirable that every student become expert in Computer Science. These courses should engender in students an appreciation of computing both for the intellectual challenges and rewards its study provides, and for the remarkable impact of its applications. Instead, these courses have often served to confirm student assumptions that Computer Science is just about programming, and that programming is a difficult and esoteric enterprise.

Traditionally, first year courses in Computer Science are divided into two categories: those that are considered as terminal or service-oriented (aka CS0 courses); and those that are perceived as introductions to the discipline intended for potential majors (aka CS1 courses). Sometimes the CS1 course is also required for mathematics and science majors, adding a service component to its mission. Most of the courses currently offered in both categories have serious shortcomings.

Many CS1 courses focus heavily on teaching the skill of programming. Since programming is an essential skill for more advanced study in Computer Science, this may seem like a reasonable approach. We must recognize, however, that many of the students who complete a CS1 course will not ultimately major in Computer Science. Thus, an introductory course focused purely on programming may offer little of lasting value to the majority of those who take it. This is not just
Recommendation 1: Most first year courses for computing majors and non-majors are in need of radical redesign; the implications of such a redesign may ripple well beyond these introductory courses. At a national level, funding is needed for curricular experiments whose results and artifacts can be leveraged across institutions. Strong departmental support for faculty engaging in this activity (both as curriculum designers and as instructors) must be provided.

With these concerns in mind, we call upon the Computer Science community to radically rethink the entire first year Computer Science curriculum. The first year curriculum should serve as an inviting and engaging experience that generates excitement about the field. Departments should be encouraged to create and offer a broad range of introductory courses. Each course should be designed both with the needs of future majors and those of the wider student population in mind. Programming should be introduced, but not as the sole or even primary focus of such courses. Introductory courses should incorporate the exploration of applications of computing that will excite students, answering the question “How does computing affect my world?” At the same time, these courses should introduce students to the fundamental threads
that interconnect the various aspects of our discipline, and should show how these conceptual foundations support interesting applications. All such introductory courses should be considered equivalent for further advancement into the field. We must recognize that such a transformation is vital but difficult, and may have ramifications that spread through the more advanced courses in the curriculum.

In undertaking an effort to revise the introductory curriculum, we must recognize that there are significant challenges in developing the types of courses we have proposed. The idea of developing a broader introductory Computer Science curriculum is not new. The fact that such courses have not already emerged is not due to lack of interest, but rather because they are inherently difficult to develop. Unlike Mathematics, Chemistry, Biology, and Physics, we cannot make the assumption that our students arrive with the basic skills required to study our field. It is much easier to present the interesting aspects of a discipline to students who already know the basics, than to present them at the same time as the basics. We recommend moving away from a product-oriented (Java, C++, C#, etc.) or methodology-oriented (objects-first, rigor-first, etc.) design of introductory courses to a broader scope of material, as discussed above. Great care and effort will be required to develop approaches that work and materials to support these approaches. We also must be cognizant of the needs of students from other fields of science who take CS1 to learn about computing.

Accordingly, the effort to rethink and redesign the introductory computing curriculum will require explicit support. Funding for the development of such initiatives should be a priority. Departments must recognize the need to encourage talented faculty members to engage in the effort required to design effective introductory Computer Science courses and must reward these efforts as significant contributions to the field. We must further recognize the importance of having our most talented and most inspiring faculty teach these first year courses, and reward these instructors appropriately. Efforts should be made to encourage dialogue among those engaged in such curricular efforts and to facilitate the sharing of the educational artifacts and materials produced.

2.2 The Bare Essentials and Enduring Fundamentals of Computer Science

We now turn our attention to the broader Computer Science curriculum. Again, the challenges are many:

- We are faced with a rapidly-changing and ever-increasing body of Computer Science knowledge. There are simply more good and valuable topics today than we can possibly teach to our majors. Also, the technical landscape will look very different to our graduates 10 years from now than it does today.
- We hear, particularly from industry, that there is an increased need for project-based and interdisciplinary courses for our majors and an increased need for students who can think systemically.
- As Computer Science concepts and methods find increased application in other disciplines, we are faced with an increasing number of students who are interested in the application of these ideas within another discipline. These students may take only a small number
Computer Science courses. It is crucial that Computer Science engage these students and practitioners, because computing — as a technology embodied in many artifacts, as a toolkit for integrating massive datasets, and as a way of solving problems — has the potential to transform a wide range of disciplines. It has already made marked transformations in several of the natural sciences and is impacting the social sciences as well. Increasingly, practitioners of non-computing disciplines must understand the power that computing ideas and techniques bring to bear, without necessarily becoming experts in computing technology. Contextualized, experiential engagement with computing is likely to increase both the transference of computing ideas and interest in computing per se.

The responses to these challenges will be many and varied. However, we believe that a common, critical step in meeting each of these challenges will be to identify the bare essentials and enduring fundamentals of our discipline. We cannot simply keep adding material to our curricula. In order to integrate disciplines, we must reduce the core of computing. In order to contextualize content, we must make room in our curriculum. In order to make our curriculum more project-based, we must cede some control of curricular coverage. To accomplish all of these things, to move computing into its next phase of development, we must strip our discipline down to its bare essentials – the core skills, concepts and methods that underlie our field. With this foundation, one can subsequently build in various directions with content of particular interest.

**Recommendation 2:** Faced with an ever-increasing body of Computer Science knowledge, an increased need for project-based and interdisciplinary courses for our majors, and an increasing number of students who can take only a few Computer Science courses, we must recognize that it is not possible for us to teach everything a computer scientist (or computing professional) might need to know. We cannot keep adding material to our curricula but instead must strip our discipline down to its bare essentials – the core skills, concepts and methods that underlie our field. With this foundation one can subsequently build in various directions with content of particular interest.

In identifying the bare essentials and enduring fundamentals of any discipline, one needs to understand 

(i) the kinds of questions it cares about, 

(ii) the kinds of results it accepts as answers, 

(iii) the methods it prefers to seek the results, and 

(iv) the kinds of evidence it accepts to validate the results. For us to provide an education of lasting value, we must teach these enduring principles complemented with current content. Our students will need to develop skills in computational reasoning and analysis, master concepts that are as foundational to the discipline as the notion of a cell is to biology, and practice the methods of the discipline to appreciate the impact of our discipline on society and culture.

When considering whether to include a particular topic or course in our curriculum, we must not ask whether the topic is good or valuable. Rather, we must ask whether it is the best possible use of our students’ time to learn it from us; that is, we must recognize that for each topic we introduce, there are other topics that we cannot introduce, or techniques that we cannot adopt, or explorations in which our students cannot engage. There are simply more good and valuable topics than our curriculum can possibly contain.
Rather than trying to teach everything that a computer scientist (or computing professional) might possibly need to know, we should concentrate on teaching only the material required for our students to learn the rest. We know that our students will need to learn material on their own, so we should stop trying to teach them everything they might need before they graduate. Rather, we should equip them for life-long learning, making this a skill that they practice even while students under our guidance. We cannot let the need for topic X, whatever X may happen to be, pre-empt the need for our students to become independent learners.

But what then are the bare essentials and enduring principles of our discipline? While the topic itself was beyond the scope of our workshop, participants noted that a recent study sponsored by the NRC Computer Science and Telecommunications Board [CSTB04] is aimed at capturing the essential character of Computer Science. Computer Science is the study of computers and what they can do – the inherent powers and limitations of abstract computers, the design and characteristics of real computers, and the innumerable applications of computers to solving problems.

- Computer Science involves symbols and their manipulation
- Computer Science involves the creation and manipulation of abstractions
- Computer Science involves the creation and study of algorithms
- Computer Science deals with artificial constructs, notably unlimited by physical laws
- Computer Science exploits and addresses exponential growth
- Computer Science studies fundamental limits on what can be computed
- Computer Science addresses the complex, analytic, rational action that is associated with human intelligence.

These ideas should shape our curriculum designs. The ideas are much more important than the specific artifacts, methods, and skills that we use to concretize the ideas. Many different examples can illustrate the above principles. To illustrate just two of those points, consider variations on the theme of sunflowers.

- **Computer Science involves symbols and their manipulation.** Computer Science deals in discrete information, with an emphasis on symbolic representation. So, for sunflowers, symbolic representations include:
  - Van Gogh’s “Sunflowers” digitized, compressed
  - Sunflowers distinguished from marigolds via genetic code
  - Sunflowers animated via a mechanical-parts representation
  - Varieties of sunflowers described and distinguished in English text
  - Nutritional values of sunflower seeds expressed in database entries

  However, for none of the above representations, does it make sense to divide its bit representation by 3.

- **Computer Science involves the creation and manipulation of algorithms.** Algorithms carry out operations that are meaningful for the intentions (abstractions) associated with
particular representations. They implicitly involve interpretation of the bit sequences that
underlie the representations. So, for sunflowers, algorithms for the above representations
include:

- Crop, manipulate images; recognize similar images
- Find locations of paintings in a database
- Seek common sequences in genetic code
- Re-position parts to create illusion of wind
- Search text descriptions for common terms
- Create nutritional labels for packets of sunflower seeds

Education, especially undergraduate education, should be of enduring value. It should prepare
the student with specific knowledge, as well as the ability to apply this knowledge in new
situations, to extend the knowledge as the field changes, and to participate knowledgably as a
citizen. The Carnegie Mellon University approach to this mission [MShaw0305], is paraphrased
here:

> A Carnegie Mellon undergraduate education aims to prepare students for life and
> leadership. In a continually changing world, the most important qualities we can
> help our students develop are the ability to think independently and critically, the
> ability to learn, and the ability to change and grow. As future leaders they must
> have courage to act, be sensitive to the needs and feelings of others, understand
> and value diversity, and honor the responsibilities that come with specialized
> knowledge and power.

We believe that this philosophy should be applied to computing as a discipline, particularly the
emphasis on fundamentals as a basis for life-long learning for our majors and on intellectual
breadth for non-majors studying computing.

### 2.3 Pedagogies and Learning Environments

The previous two sections have focussed on the content, organization, and management of the
curriculum. Workshop participants also discussed the need for a radical rethinking of not only
what we teach but how we teach it. The need for changes in pedagogical approaches must be
addressed on multiple fronts. In the classroom, updating course content alone is insufficient.
New teaching methods — such as active learning, collaborative learning (e.g., pair
programming) and problem-based learning — must be supported. Experiential and contextual
learning environments that allow students to customize their experience to their personal learning
style are needed. Technical instruction must be integrated into a broader social context.

While some Computer Science faculty have already begun experimenting with these ideas,
workshop participants called for the community to focus on both content and pedagogy. Richard
Felder has written on his home page [RFelder]:

> College teaching may be the only skilled profession for which no preparation or
> training is provided or required. ... The result is the consistent use of teaching
techniques that have repeatedly been shown to be ineffective at promoting learning. Many professors are surprised to learn that...

i. There are well-defined instructional techniques that make teaching more effective.

ii. These techniques can be introduced slowly and methodically, without compromising coverage of the syllabus. They do not require large expenditures of money, time, and effort.

iii. Most importantly, the techniques have been validated by careful, documented, repeatable research. Their effectiveness is not simply a matter of opinion. They work!

General education literature notes that different students have different cognitive learning styles. Relatively few learn best from the traditional lecture method. For many students, active learning building on and correcting pre-existing concepts has been shown to be successful. Students learn better when all their senses are involved in tasks that they feel are relevant to them or to the good of their world. We do not have the space or expertise to cover this topic in depth here; some good sources are [BBC00], [RFelder], [MargolisFisher01], [RodgersStarrett05] and [Zull02].

Although Felder asserts that "the techniques have been validated ....," we are not aware of their having been validated in Computer Science. There is good reason to suspect that techniques that work well in other sciences will work well in Computer Science. But it is also valuable to look beyond the sciences. For example, the classic pedagogy in studio art is based on a studio model with formal critique; such an approach may be well-suited for digital media studies. We think there is a need to support research into educational strategies and techniques specifically for Computer Science.

When considering our pedagogy, we must consider the student of tomorrow. One such student is described by Rodgers and Starrett [RogersStarrett05] as:

He sits at the computer with headphones piping music from an iPOD to his ears. Ten different MSN chat windows blink and chime on the computer screen. An online role-playing game is minimized on the Windows taskbar. A music video blares from a TV in a corner of the room. A calculus book lies nonchalantly open by the cell phone, which itself sits next to the PC. He is doing his homework. He is real. He is a 21st Century Learner.

Alternatively, Werner [CRA 2005d] begins her article on “Want to Increase Your Retention of Female Students” with the following scenario:

A nerdy-looking guy sits alone working at a computer late at night. Is this a portrait of your typical Computer Science student? Or instead, does your typical student look like one of a pair of students working together at one computer—laughing, talking, pointing to the monitor, looking at each other, and having fun?
Most likely, both scenarios, as well as a spectrum of learners and learning styles in between will need to be accommodated. This points to the need for both personal and group learning environments that are flexible, adaptive, and highly personalized for the learner.

The computing laboratory is a common learning environment found in many Computer Science departments. But laboratory spaces that feel like barns of PCs do not constitute an ideal learning environment, and with most college students having their own computers, the notion of a computing lab where a student goes primarily for access to a computer has become obsolete. What then should our communal learning spaces look like, and what should be their function? Computer Science might well borrow ideas from the Project Kaleidoscope (http://www.pkal.org/), whose publication *PKAL Volume III – Structures for Science: A Handbook on Planning Facilities for Undergraduate Natural Science Communities, 2005* is “intended for use by colleges and universities that are thinking about, or in the process of planning for, new or renovated spaces for their undergraduate programs in science and mathematics.”

These changes need to be made in a principled manner. The computing academic community needs to become familiar with the results of social science research investigating what works and doesn’t work in a learning environment. It is absolutely crucial to disseminate relevant studies to the academic Computer Science community. ACM President David Patterson has suggested using *ACM TechNews* to report such results to the community [DPatterson1105]. Other avenues of dissemination include invited speakers at conferences such as the *ACM Federated Conference on Research in Computing (FCRC)* and the annual *ACM SIGCSE meeting*. We need to adopt an Open Source model for pedagogical materials and study results, establishing *Course Forg.net* to make sure they are accessible throughout the community. We need to organize teaching workshops for our faculty, particularly recent Ph.D. graduates. The ACM SIGs could offer such one-day workshops collocated with their annual meetings, as has been done at ACM SIGCOMM [Kurose 2002], or could fashion teaching workshops similar to existing academic mentoring workshops, such as the New Software Engineering Faculty Symposium held at ICSE by ACM SIGSOFT and IEEE-CS TCSE.

The academic computing community must hear from the pioneers — researchers such as Jane Margolis and Joanne McGrath Cohoon — whose successes in studying effective teaching methodologies for women students have been reported recently. An important outcome of such efforts would be to open channels of communication between the social science and computing communities to support research that develops sound pedagogy and learning environments that enhance learning.

Yet another challenge is to integrate experiential-based, socially-responsible collaboration into the curriculum. Problems that address social responsibility can go far toward making introductory Computer Science both more relevant and accessible for a range of students and
toward helping to combat our discipline’s Dilbert-like image (see Section 4.3). For example, students in an undergraduate database course can implement a solution for a non-profit organization. Students in a networks course can solve a security problem for an elementary school. Any introductory course that includes implementing a game can extend the requirements to specify a game appropriate for a particular cohort, for example a math game for children with special needs.

In proposing curricular innovations to introductory courses (see Section 2.1), a concern is that such innovations are unlikely to be aligned to Computer Science education theory, since this theory is not sufficiently mature to answer many important questions. Questions such as those concerning students’ mental models of objects and object-oriented programming, relations between a student’s style of learning and the style of material presentation and/or the assignments best suited to that student, and even how to assess mastery of content are all valid Computer Science education research questions to which we do not yet have sufficient answers. Computer Science education research does not have the respect of mathematics education research or science education research. Better funding of Computer Science education research would allow curricular and technical innovations to be better tied to the theory. Additionally, providing such funding will help convince Computer Science research departments of the validity of such research. Respected programs, such as the Carnegie Academy for the Scholarship of Teaching and Learning, espouse precisely this concept of the study of effective teaching as scholarly work [Carnegie 2005].

To make sure our teaching methodologies are designed for a broader audience, sensitivity training for administrators and faculty members in computing, similar to the training that was necessary when women first started entering science and engineering majors in large numbers, should also be considered. Highly respected academics such as Richard Tapia and Bill Wulf could be enlisted for this purpose.

We need to make every computing course add value to a student's overall education whether or not they intend to work in the computing field. At least part of our task should be to determine how we can accomplish this. How do we make every computing course interesting and exciting? By this we do not mean that every hour has to be exciting, but that by the end of every course, students should appreciate the value of the course; they should not have to wait 2 years to appreciate it. How do we educate students so that they can 'catch-on fast' or (according to the CISE executive summary describing the aims of this workshop) how do we "produce graduates who are intellectually agile in a dynamically changing discipline"?

2.4 Research versus Education: Promotion, Tenure, Funding

An expressed concern among workshop participants was the relative lack of importance attached to teaching and related educational issues by many academic institutions, particularly with respect to tenure and promotion. Many felt that the lack of importance attached to teaching was a potential show-stopper in making effective changes to Computer Science curricula and pedagogy. Some workshop participants felt that the problem is particularly acute at universities striving to improve their national ranking as research institutions. While recognizing that there are as many variations as there are institutions, the identified problem exists in many places.
Often emphasis is placed on research-related measures, such as number and quality of publications and number/amount of grants. Teaching, whose results are much harder to quantify, undervalued. Great teachers have not been able to get tenure and/or promotion without the requisite research credentials, even if they are highly-innovative and well-respected. Moreover, there is often a lack of support for those interested in, and desiring to work on, necessary curriculum change. Overall, the relative importance given to research versus teaching by P&T committees is significantly skewed towards research.

The consequences of this situation in the academic community are that:

- younger, tenure-track faculty are not sufficiently involved in course and curriculum issues,
- recently-tenured faculty are not encouraged to devote time and energy to curriculum development, and
- insufficient recognition is given to the importance of designing truly innovative introductory courses that will not only attract, but help retain students, majors as well as non-majors (as discussed in Section 2.1).

In short, we believe that educational institutions must recognize the importance and the difficulty of developing and teaching introductory courses and continuously updating curricula in computing. These issues are crucial to our discipline and to our research. Indeed, without effective teaching, we will not have the quality students who power our research engine. We must provide commensurate resources and recognition for the people involved in these activities. The combined voices of professional bodies (e.g., IEEE Computer Society, ACM, National Academies, NSF panels, CRA) can help effect this change and have significant impact. For example, the 1994 NRC report, Academic Careers for Experimental Computer Scientists and Engineers [NRC 1994], has significantly influenced how promotion and tenure committees view the relative importance of journal versus conference papers in the tenure and promotion process. It is important that this effort be broadly based across our discipline (i.e., including four-year colleges as well as the Ph.D granting institutions in North America that are members of CRA).

The tensions between research and education in academia are reflected in similar tensions within the funding agencies. With constrained budgets, hard decisions must inevitably be made. Programs that had previously been targeted specifically at education (e.g., NSF Education Innovation and NSF CRCD) appear to be dormant. Just as universities must be called on to emphasize the importance of educational activities (i.e., curriculum development, pedagogical excellence, and the scholarship of teaching) and to reflect this emphasis in their reward structure, so too must the funding agencies. Needed education innovations can occur only with both universities and federal funding agencies fully on board.

**Recommendation 4:** Innovative teaching and curriculum development must be valued as significant faculty contributions and achievements, and rewarded as such by P&T committees. Professional bodies such as IEEE Computer Society, ACM, National Academies, NSF panels, and the CRA can help effect this change. It is important that any such activities involve and address not only Ph.D. granting institutions, but 4-year colleges as well. Just as universities must be called on to emphasize the importance of educational activities and reflect this emphasis in their reward structure, so too must funding agencies. Needed education innovations can occur only with both universities and federal funding agencies fully on board.
3. Looking Outward: The Computer Science Curriculum Broadly Interpreted

As Computer Science matures and evolves as a discipline, it must continuously redefine not only what it is but also what it wants to be. A primary challenge arises from the ever-increasing breadth of computing, which is infusing and transforming our academic disciplines (not only within science and engineering) and ubiquitously permeating our work and our culture. What will be the relationship between our discipline and these other areas?

• How will Computer Science relate to and cooperate with closely associated disciplines (i.e., Computer Engineering, Software Engineering, Information Systems, Information Technology) that themselves have existed for decades?

• How will Computer Science relate to “application” areas (i.e., Informatics, computational sciences, electromechanical and embedded systems, learning sciences, system sciences, social sciences, management, arts and humanities)?

• How will Computer Science play a role as a fourth fundamental (i.e., reading, ’riting, ’rithmetic, and ’rithms) for a well-educated population, and/or as a new mode of scientific research (i.e., computation, simulation, or information-driven research as a complement to theory and experimentation)?

Will Computer Science narrowly define its scope to be the core, highly-technical areas that have defined it for decades, or will there be a more expansive view? The workshop participants felt strongly that Computer Science should take an expansive view of its domain and strive to be viewed as both broad and fundamental. Excluding new areas and applications will not serve us well in the long run. The advantages to an expansive definition of Computer Science are many.

• Computer Science has much to offer to other disciplines, and it can often shape and transform disciplines. Computational paradigms have changed the core of many disciplines and enabled new kinds of questions. For example, Computer Science has added a new in silico paradigm to the sciences’ traditional in vivo and in vitro research paradigms.

• The workplace of the 21st century will be knowledge- and computing-intensive, demanding fluency and often requiring a deeper understanding of computing theory and systems. The workplace will also value those intellectually agile, multi-disciplinary workers who can cross boundaries between units and organizations. A multidisciplinary view of computing reflects the realities of current organizational practice well, demonstrates the importance of multiple perspectives, and increases the credibility of the field among industry practitioners.

• Adopting a broad view of computing and nurturing cooperation between computing disciplines allows all computing disciplines to serve their stakeholders better by building on not only their own strengths but on the strengths of the other disciplines. For example, Information Systems can benefit from the computational and algorithmic expertise of Computer Science while Computer Science can rely on Information Systems for issues
related to the modeling and representation of complex organizational/business requirements. Models of organization and resource sharing in biological and social systems can inform the design of similar mechanisms in engineered computing systems. Additionally, departments of Information Systems and Information Technology have (as have traditional departments of Computer Science) been educating computing students for decades. All sides can benefit from the special expertise of the other.

- A broad view of computing increases its relevancy. The danger of narrowly circumscribing our intellectual foci is diminished relevancy. It is telling that while many Computer Science departments have seen a decrease in enrollments, schools of Information Technology (that embody a broader definition of computing) have not experienced such declines.

**Recommendation 5:** Computer Science should take an expansive view of its domain. As a discipline, Computer Science should be viewed as both broad and fundamental. Computer Science can often shape and transform other disciplines as well as learn from them. We would do well to include but also look beyond the traditional science and engineering disciplines, as there are numerous opportunities in the arts, humanities and social sciences. An expansive view of Computer Science also fits well with the vision of a 21st century workplace that is knowledge- and computing-intensive, demanding some minimal fluency and often requiring a deeper understanding of computing theory and systems.

Software Engineering. A number of Computer Science departments have embraced areas as robotics and human-computer interaction as being within our scope and have developed computational science courses that build bridges to the physical and life sciences. Other Computer Science departments have computational biologists and applied mathematicians. We believe Computer Science should do well to include but also look beyond its sister disciplines in science and engineering. In particular, numerous opportunities in the arts, humanities, and social sciences can not only benefit tremendously from an infusion of Computer Science but also can lead to deep Computer Science research. Encouraging close collaboration will make clearer to other disciplines and to society at large, the many benefits that a computing perspective can confer.

### 3.1 Challenges in Instantiating an Expansive Definition of Computer Science

Cross-disciplinary research and education are difficult to initiate because they involve efforts to traverse established boundaries and reward systems. Federal investments in research programs such as ITR and multi-disciplinary research centers (e.g., NSF STCs and ERCs) provide
important incentives and opportunities for collaboration. Translating transformative research that crosses traditional boundaries and revolutionizes our way of approaching problems into effective courses, curricula and programs can be difficult. There are a number of challenges:

- First and foremost is the question of resources. These areas are sufficiently young so that there are no good textbooks or developed educational materials. Researchers invariably drive the development of novel and interdisciplinary courses, curricula and programs. At the same time, it is these researchers who are key to the success of the research. Both time and funding resources are critical in developing new educational materials.

- Secondly, when faculty members in Computer Science teach material outside the traditional core of the discipline, it puts more pressure on the rest of the department to cover that traditional core. This cost is often hidden from view, but it needs to become part of the planning, accounting and funding process.

- A third challenge involves the dissemination, adoption, and standardization of curricula. Replication, especially in interdisciplinary areas, is very important. Replication is complex because it is difficult to capture the circumstances and environmental constraints that enable success in the driving research programs. On the other hand, an early emphasis on replication and standardization may constrain an already difficult development process and thwart innovation. Here, funding models can make a real difference: for instance, a model with a first round of seed funding to innovate, followed by a second round that supports projects explicitly emphasizing achievement of widespread adoption, dissemination and evaluation, may prove successful.

- A final challenge is that of mindset. Unfortunately, some people in other disciplines still see Computer Science as a handmaiden or a mere provider of solutions; they often convey the attitude, “We have the ideas, you write the programs for us.” Both in our own educational offerings and in our collaborations on interdisciplinary offerings, we should insist on incorporating the grand ideas of computing, not just the skills often taught today in service courses (see Section 2.1).

3.2 Approaches to Cross Disciplinary Education and Teaching

There are several different models for cross-disciplinary education and teaching. These include:

- **Introducing cross-disciplinary material in the Computer Science curriculum**, that is, developing and offering Computer Science courses that incorporate significant non-traditional content meant to endow students with non-trivial domain knowledge (see Section 2.3 of this report on pedagogy).

- **Creating computing modules that other disciplines can**
incorporate into their curricula, or co-teaching cross-disciplinary courses that may be derived from interdisciplinary research projects. Our greatest contribution to integrating Computer Science with other disciplines will be our unique mindset: our conceptual base, our style of reasoning, and our values. The key to integrating Computer Science with other disciplines is helping students understand the mindsets of both disciplines and the ways in which they interact. Our purpose should be to identify content that is modern and appropriate, techniques and organizations for teaching it, and approaches to merging it into interdisciplinary offerings. This can be done by:

- insisting on incorporating the grand ideas of computing (see Section 2.2 of this report; see also [Denning 2005]), not just the sorts of skills that wind up in service courses,
- showing how Computer Science contributes a full suite of distinctive intellectual tools, not just programming,
- developing an inventory of domain-specific examples that demonstrate the characteristics of Computer Science, and
- showing concretely how the ideas have affected the partner discipline.

In developing interdisciplinary offerings, Computer Science should be a full intellectual partner. Just adding service courses is not true integration. Service courses, properly conceived (see Section 2.1) have their place, but they don’t lead to a true interdisciplinary partnership.

- Encouraging the introduction of computing skills, concepts, and capabilities in other disciplines, (e.g., IT fluency and IT across the curriculum, [CSTB 1999]).

- Creating core computing curricula, as a basis for:
  - integrating the various computing disciplines (Computer Science, Computer Engineering, Software Engineering, Information Systems, and Information Technology),
  - providing the technical fluency for students to combine with their own disciplinary studies, and
  - combining with cognates to form interdisciplinary educational programs.

**Recommendation 7:** A “one-size fits all” model of computing education is not sustainable in the long term. The curriculum must allow for a “multiple entry / multiple opportunities” approach that allows students to take many different paths through an expansive computing curriculum. We must be able to accommodate students with diverse backgrounds (multiple entry points), diverse interests (multiple pathways through our major) and diverse educational goals (multiple exit points from our major).
Recommendation 8: Several organizational models are evolving that bring a variety of computing- and information-centered disciplines (e.g., Computer Science, Computer and Information Science, Library and Information Science, Information Systems, Informatics, Information Technology, Information and Communications Technology) as well as application areas under a single organizational unit, similar to engineering and business schools. We recommend that an agency such as NSF or the CRA and its Information Technology Deans Group undertake an effort to lead a community discussion of these various approaches in forms of a careful study of alternative organizational models for schools that bring together computing- and information-centered disciplines under one “roof”, as well as a meeting of heads of such departments in arts and sciences, engineering, and business schools (and their deans) to discuss content, organizational and structural issues.

3.3 Organizational Models for Computing in the Academy

In North America, the names of academic units that house computing faculty include Computer Science, Computing Science, Computer Engineering, Computer and Information Science, Library and Information Science, Information Science, Information Systems, Informatics, Information Technology, Information and Communications Technology (and these words can occur as plural – e.g. sciences, technologies). In Europe, Informatics seems to be used synonymously with Computer Science in Germany and France, used more broadly in Holland, Scandinavia, and Italy, and most broadly in the U.K. with information processing systems, artificial or natural (cognitive science). Information technology (IT) is often used as a generic term for all of these, as we are using it in this report.

There is a technical dimension to this set of programs that can be decomposed into computing, information, and engineering components, with engineering decomposed into electrical and computer engineering. There is a human dimension that includes human computer interaction, social and organizational informatics — the role IT plays in society and within organizations. There is a domain dimension, including Bioinformatics, Chemical informatics, Laboratory Informatics, Health Informatics, Music informatics. There are more traditional domain applications of IT such as (i) Computer or Management Information Systems, in most cases in business schools and now sometimes in IT schools, and (ii) Library Science, sometimes in Schools of Library Science, sometimes now in Library and Information Science, or in Information Schools. There is a design dimension to all or most of these programs.

Most computing programs do not have all of these dimensions, but they might position themselves by choosing the dimensions that they include or emphasize. The real challenge for
computing is to be *trans-disciplinary* while remaining rooted in the traditional computing disciplines.

Several organizational models are evolving for schools of IT:

- Departments (Computer Science, Information Science, Information Technology, Informatics, etc.); traditionally these are rigidly organized with separate chairs, curricula committees, degrees, budgets.
- Interest groups or “faculties” (Software Engineering, Database, Networking); leaders here might be directors, group leaders, program chairs.
- Group of the whole; leaders might be associate deans.
- Levels (core, advanced, masters, doctoral).

A School of Information Technology provides enhanced visibility within a university, both internally and externally. In principle, it is more efficient by avoiding redundancy in programs, projects, and administration. It exists “on a level playing field” with other schools and has enhanced access to the executive administration and to external constituencies. It brings together in one place faculty expertise in computing areas for education and research. These advantages combine to bring about the following benefits:

- Informed promotion and tenure procedures in the generally less known areas of computing
- Increased interaction with other schools in the university
- Enhanced internships and placement
- A strong industry advisory board
- Better literature and promotional material
- Stronger student recruiting
- Strengthened alumni/ae affairs
- Stimulus to innovation, entrepreneurship
- Enhanced funding opportunities: private, corporate, governmental

But IT schools have challenges, including:

- Integrating Computer Science/ Information Science/ Information Technology/ Informatics faculty
- Enabling students to “try out” different courses/programs within a school of IT
- Avoiding redundancy of courses, programs, projects
- Eliminating unhealthy competition among programs

While there seems to be an emerging consensus that Computer Science should be broadened and integrated with other disciplines, the challenge is how to do this without Computer Science being absorbed into other disciplines and being perceived as only a tool. Mathematics is one model to consider. Although Mathematics is ubiquitous, it has kept its own unique identity. But mathematics also tends to be isolated, and Mathematics departments have a tendency to push out the more applied aspects of mathematics, such as Applied Mathematics, Logic, and Statistics. Better examples might be business and engineering schools, which have their own niche in
academia but have also become multidisciplinary in many respects. Medical schools might be an even better paradigm. No one seems to mind much if a medical school has a Biochemistry department, or even a biomedical ethicist. Small colleges might include subjects that relate to “pre-med,” (e.g., Biology and Chemistry). One would expect the same with IT. A small college might have a Computer Science department that might even be multidisciplinary to some extent, but one would not expect to find as many dimensions of IT as at an IT school of a major research university. While there is not a recommendation here for Schools of Information Technology, there is for the breadth and integration concept; an IT department might well span more than Computer Science; it would then enjoy some of the advantages listed above and share some of the challenges.
4. Broadening Participation

The Computer Science community must consider the issue of computing education for the 21st century from the perspective of how to serve the broad population of educated Americans. Recruitment and retention are clearly crucial issues, with declining enrollments in the Computer Science major and a particularly alarming decrease in the number of individuals from under-represented groups majoring in Computer Science. While some of the decreases may be ameliorated by new curricula and improved pedagogy, these do not address our inability to draw students into our discipline at an early age in the K-12 pipeline.

In this section on broadening participation in the computing discipline, we consider each of the following issues in turn:

(i) problems in the K-12 pipeline
(ii) achieving a more diverse population in computing
(iii) improving the image of computing
(iv) improving the educational linkages between academia and industry

The need to broaden the population in computing disciplines is clearly reflected in the graduation rates in the 2004 Taulbee data from Computer Science/Computer Engineering Ph.D. granting institutions in North America [CRA 2005e, CRA 2005f]. These reports show the alarmingly small numbers of women and underrepresented minorities graduating with degrees in our discipline. In 2004, only 17% of the bachelor’s degrees, 25% of the master’s degrees and 18% of the PhD degrees were awarded to women. Depressingly, these numbers have changed little since 1993, when CRA began gathering this information. The situation for underrepresented minorities receiving degrees in CS&E in 2004 is even worse:

<table>
<thead>
<tr>
<th></th>
<th>Bachelor’s</th>
<th>Master’s</th>
<th>PhD</th>
</tr>
</thead>
<tbody>
<tr>
<td>African-American</td>
<td>3.4%</td>
<td>1.6%</td>
<td>1.5%</td>
</tr>
<tr>
<td>American/Alaskan Native</td>
<td>0.4%</td>
<td>0.4%</td>
<td>0%</td>
</tr>
<tr>
<td>Asian/Pacific Islander</td>
<td>22.9%</td>
<td>16.9%</td>
<td>12.8%</td>
</tr>
<tr>
<td>Hispanic</td>
<td>3.9%</td>
<td>1.3%</td>
<td>1.1%</td>
</tr>
<tr>
<td>White (non-Hispanic)</td>
<td>54.5%</td>
<td>50.6%</td>
<td>34.7%</td>
</tr>
<tr>
<td>Nonresident Alien</td>
<td>10.2%</td>
<td>2.9%</td>
<td>48.2%</td>
</tr>
<tr>
<td>Other</td>
<td>4.9%</td>
<td>2.9%</td>
<td>1.7%</td>
</tr>
</tbody>
</table>

There is also concern about the numbers of high school students taking AP Computer Science classes. Students taking high school AP classes often choose to major in the subject in which they have taken an AP course. Despite a 33% increase in students taking AP tests overall (from 2001-2004), the number of students taking the AP Computer Science exam was down 9% in the A test, and 20% in the AB test, with Computer Science also evidencing the largest gender gap. The decline in Computer Science AP participation may also reflect a decline in exposure to Computer Science in grades K-12. Whereas Curriculum ’78 [Curr78] assumed that by 1990 all well-educated high school graduates would have a rudimentary knowledge of programming upon
entry to college, the reality 25 years later is that some high school students have fewer opportunities to study Computer Science concepts.

4.1 The Pipeline Problem: K-12 Exposure to Computing

Attracting students to computing as a discipline must occur prior to their entering college. Students need to be shown computing careers in a positive light, starting in middle school. This should occur in curricular as well as extracurricular settings. There is a need to address the lack of specific computing and information science topics in the K-12 system as a whole. Pre-college students need to understand the skills needed for career advancement in the 21st century. Young Americans need to see themselves as creators of high technology, not just consumers. They should be exposed to computing professionals at their high schools; providing such speakers may be a goal that our computing professional societies can adopt. Various misconceptions among students as to the availability of computing jobs need to be addressed. The web-bubble bust has created the impression in the general public that computing jobs are no longer available in the US; massive off-shoring of programming and information technology jobs is also assumed.

In elementary school, children should use computers creatively to construct artifacts, not simply as tools to support other subjects (e.g., researching a Social Studies topic, or modeling a data collection problem). Logically this leads to the question of how the current K-12 teaching cadre can be trained to provide truly computing-centric instruction. The creation of the Computer Science Teachers Association (CSTA) by the ACM has been a positive first step toward developing national standards for teaching K-12 computing. Support will be needed to implement CSTA guidelines and frameworks into courses that can then be replicated across school districts. The guidelines, training, and in-school support need to focus on how foundational concepts of computing impact the way in which we navigate the world. CSTA members may provide the means for computing professionals to visit their schools to talk to students. By hosting such visitors, the CSTA members will be providing role models in computing for their students, and the ensuing discussions may help to combat the misconceptions noted above.

Exacerbating the problem of inadequate exposure to the concepts of computing in K-12 is the lack of explicit state and national educational standards in Computer Science. In this era of educational accountability, the material that is in the standards is the material that will be taught. The emphasis on testing and accountability may partially explain why there is a decline in the Computer Science AP course enrollment — in order to excel in high school, a good student may not be able to find time for this course. Until Computer Science appears explicitly in state standards, an over-burdened K-12 curriculum is unlikely to find a meaningful place for computing topics.

A formal assessment of computing in state standards is needed. For example, New Jersey state standards are fairly typical of those established throughout the country. A careful reading of standards shows a complete absence of reference to *Computer Science, information science, or programming*. However *algorithms, data, problem solving, problem analysis, solution design* are all topics dispersed throughout the standards under a diverse set of topics ranging from...
Mathematics to Social Studies to Language Arts. Students who take to a particular topic (e.g., problem solving and algorithm development) may not realize that these topics are fundamental to Computer Science. They will not see the connection between their facility at problem definition and solution, and those computing careers that exploit these skills.

**Recommendation 9:** Resources for K-12 teacher training in computing need to be made available. Standards for K-12 computing education must be enacted as a national educational priority, facilitated by a meeting sponsored by NSF and the National Academies. Educators, representatives from business, and (most importantly) state education commissioners would be key participants in such a workshop.

In order to affect standards at the state level, we must involve state commissioners of education in the effort. These are the individuals who can make K-12 standards for computing education happen, and no broadly-based effort to integrate Computer Science into the K-12 curriculum will be successful without their support. We can also enlist the cooperation of our professional societies. We recommend that NSF and the National Academies sponsor a workshop including educators, representatives from industry, and (most importantly) state education commissioners, to facilitate creation of standards for K-12 education in computing.

Another concern is the quality of teaching of computing concepts in the K-12 level (see Section 2.1 for a discussion of the quality of teaching introductory computing courses at the college level). As in the case of introductory college-level computing courses, K-12 courses must engage the interest of a broad cross-section of the population, engendering an appreciation of computing both for the intellectual challenges and rewards its study provides and for the remarkable impact of its applications. Professional development opportunities in computing should also be provided to teachers (especially high school pre-AP and AP teachers) to help them expand their own knowledge and create effective classroom lesson plans and materials for classroom use. Guidance counselors also play an important role in informing high school students about the benefits to students of studying Computer Science in college.

If the schools do not have the space, time, or motivation to expose young people to Computer Science, extra-curricular venues become even more crucial. Extra-curricular, after-school computer clubs can be created and supported. Groups such as the Girl Scouts should be encouraged to provide opportunities for exposure to computing. Summer camps, where students are exposed to computing, can also provide positive computing experiences. It would be especially beneficial if computer experiences during summer camps somehow could be connected to in-school learning during subsequent school years.

Colleges can work with middle and high schools to excite the students about computing. Mathematics departments have been successful in running events such as Sonya Kovalevsky Days for girls [AWM 2005]; Computer Science departments could run similar events. Activities such as the Carnegie Mellon University Women@SCS (School of Computer Science) Road Shows for Girls [CMU 2005] are similarly aimed at sparking the interest of middle school girls in computing topics. Visits by computing professionals during career days should be encouraged. There are also some excellent examples of successful partnerships between 4-year colleges and community colleges as part of the NSF ATE program. This collaboration will offer both
enrichment for the K-12 systems and a community service opportunity for the college students studying computing.

4.2 Appealing to Members of Under-represented Groups

Efforts need to be made to attract women and under-represented minorities to the various computing disciplines and to ensure their retention through the first year of computing courses where the majority of attrition occurs. Attracting students to computing majors is perhaps best accomplished by creating highly motivating, meaningful courses to serve as students’ first exposure to computing (see Section 2.1). These courses should be aimed at showing (at least partially) the interdisciplinary nature of computing and dispelling the idea that computing is simply programming. Specific relevant content and approaches that are sensitive to a student’s ethnicity, sex, and learning style are an essential part of any successful intervention. Mark Guzdial [Guzdial] has had great success in using multimedia as a hook to excite students about programming. Dann, Cooper, and Pausch [DaanCooperPausch05] have had comparable results in their use of the programming language Alice, enabling students to create animations and build computer games as they learn to program. Other novel approaches involve the use of graphical user interfaces (GUIs) and special-purpose graphics libraries, as well as improved development environments, such as Dr. Scheme (http://www.drscheme.org/), that appeal to today's students, who have grown up in a media-rich world.

It is important to attract members of under-represented groups to computing at the high school level. The Curie Program exists at Cornell to entice female high school students into engineering. It is quite successful, and many of the high school students end up at Cornell — and then later act as undergraduate TAs in the course. The program is so successful that the College of Engineering at Cornell is setting up a similar one for under-represented minorities [Cornell 2005].

A particular mention should be made concerning community colleges and their issues of attracting and retaining computing majors. Community colleges play a crucial role in both technical training and in feeding students into four-year colleges. Overall student attrition runs significantly higher in community colleges, with as few as 1 in 6 students graduating. Community college classes are often more challenging to teach, since there is a much wider range of student ability. Also, community college faculty are often coping with high credit-hour course loads and typically have less graduate level training than 4-year and university level faculty. Thus, community college faculty members have greater needs for faculty professional development.

Attracting a wider range of students to computing majors, while necessary, is not sufficient. Retaining prospective computing majors in introductory-level classes is also important, as the traditional withdrawal/fail rates in introductory Computer Science courses (often as high as 50%) are too high in the current environment of too few students choosing initially to major in computing. Many studies (e.g., [MargolisFisher01]) track the causes cited by students, women and minority students in particular, who have chosen to leave the Computer Science major. Efforts should be made to implement interventions to combat the conditions that cause students to leave the major.
Recommendation 10: Efforts must be expanded to recruit and retain women and under-represented minorities in computing disciplines. Needed are new introductory approaches to the discipline, the use of student-engaging teaching methods, emphasis on human-centric computing, and establishment of support structures on campus for members of these groups.

Assume that we can achieve a diverse population of undergraduates, including women and minorities, each of whom has taken at least one introductory course in computing. Further assume that these students are considering (or have already declared) a major in a computing discipline. Several actions can be taken to improve the probability that these students will successfully complete this major:

- **Establish a human-centric view of computing.** A human-centric view of computing must be conveyed that will appeal to this diverse student group. Simply put, the prevailing machine-centered and male-biased image of computing must be replaced with a human-centered view of computing as a field that is fascinating, exciting and tackles and solves innumerable problems in areas that affect – enhance and improve – people’s lives. Changing Computer Science curricula to emphasize computing embedded in real life situations would help to correct this image (see Sections 2.1 - 2.3, 4.3).

- **Employ student-engaging teaching methods.** New approaches to teaching are needed that will engage a diverse student population. The academic environment in which computing is taught must fit better with the reality of the outside computing world and must seem relevant to the students’ lives (see Section 2.3).

- **Widen the definition of the computing major.** The notion of what comprises a computing major must be enlarged to include interdisciplinary programs such as Digital Multimedia and Bioinformatics. Other disciplines must be synthesized with Computer Science; the resulting areas should then be integrated into new Computer Science curricula and used to attract and retain under-represented groups (see Sections 3.1,3.2).

- **Build support systems.** Support systems are essential if underrepresented groups in computing are to succeed in their choice of a major. A key question is, “What types and levels of support are adequate for this purpose?” Students from diverse backgrounds, especially those who may be at risk of dropping out of computing, require a solid system of support to guarantee their success. Tutoring and mentoring services should be available and easily accessible. Providing assistance in acquiring oral and written communication skills should be a high priority. The goal should be to establish support systems and services that enable under-represented groups to reach a critical mass in computing.

We believe that the percentages of women and under-represented minorities in computing will increase significantly only when the field of computing itself takes significant steps to improve the situation. To address this issue in an exciting, effective way, we propose the following 50-by-50-by-5 challenge: “For the next five years, fifty Ph.D.-granting computing departments will each hold a 3-week summer course for 25 high-school women and a 3-week summer course for 25 high-school minority students. The students will take two courses: one on programming and the other on innovations in computing. Additionally, 25 departments will hold a 1-2 week course for high school teachers to learn about teaching programming.”
Recommendation 11: A national “grand challenge” is needed to improve the percentage of under-represented minorities in computing, focused at the high school level. A 5-year 50x50x5 plan will have 50 academic departments hold a 3-week course for high school women; 50 academic departments hold a 3-week course for high school minority students; and 50 academic departments hold a 2-week course for high school teachers.

The program would attempt to excite 2,500 students each summer about computing, and we expect a fair number of them subsequently will study computing when they reach college. Moreover, if the students are excited, one can expect them to pass the excitement on to their friends.

A program of this size is an ambitious undertaking. Funding is needed for instructors—both to prepare and to give the courses—and for teaching assistants. Funding is needed for travel, room, and board for all students (and the high school teachers). In addition, some students might need a stipend in order to take part, because they will not be working. Finally, funding will be needed to administer the program. Back-of-the-envelope calculations indicate that between $6 million and $10 million would be needed each year to support this effort. We would hope that the NSF, the CRA, and the ACM would be involved and would contribute some funding. But the bulk of the funding could come from computing-related companies. If the program is publicized and marketed properly, many companies would want to contribute.

The program could be administered by one computing department. It should be overseen by the NSF and, perhaps, the CRA. Members of CRA-W and some minority computing faculty should be involved, so that the program sends the proper message to the students. We envision a process where computing departments would submit a short proposal in order to participate.

4.3 Improving the Image of Computing

Unfortunately, there is a public perception that the computing industry is populated by geeks and nerds. The image of a prototypical white male with glasses, pocket protector, and sloppy dress is not a positive image; further it signals to women and minorities that they are not included or welcomed into the computing profession. This image is exacerbated by the entertainment industry, notwithstanding the ‘good press’ of the popular TV shows CSI and Numbers, for example, which feature use of computers and computational thinking for the good of society. There is also a feeling among academic computer scientists that colleagues in academia see computing as a tool and a means to an end, and not as a true scientific discipline. In addition, our profession has not communicated with K-12 educators or guidance counselors the true nature of what computing and information technology workers do, what impact they have on the world, and how exciting, rewarding, and valuable the work is. Parents do not have adequate information to encourage or support a child’s decision to enroll in a computing program. Finally, the computing industry seems to lack sufficient positive role models who can make a compelling case on its behalf. (Kudos to Bill Gates for his recent college campus tours that were aimed at encouraging study of Computer Science.)

This is the reality. Our challenge is to transform that negative into a positive. One suggestion is to convene a meeting of stakeholders in education, industry, and government to address this image problem through a national campaign to attract students to computing. The gathering
should include representatives of national professional societies (ACM, IEEE, American Association of Artificial Intelligence, etc.), state and national industry trade associations (Information Technology Association of America, Software and Information Industry Association, Business Software Alliance, American Electronics Association, TechNet, Computer Systems Policy Project, Coalition of Regional IT Associations, etc.), as well as the public sector (Congress and Administration). The goal of the gathering would be to devise specific strategies and next steps to address this problem. We should look at what the AAAS and AMA have done to solve a similar issue with regard to mathematics.

After our workshop, we became aware of the Computing Research Association (CRA) efforts in this direction. Jim Foley, CRA Board Chair has written [CRA2005c],

The Computing Research Funding Task Force, led by CRA, will develop a coalition of societies and companies to be the source of computing research information and advocacy to the government and to coalitions such as ASTRA, the Council on Competitiveness, the National Association of Manufacturers, TechNet, the Task Force on the Future of American Innovation, as well as to our member societies. The CRA task force will aggressively present the case for computing research to the administration and the legislature, drawing on the human and financial resources of our corporate and society members for personal visits, print media, behind-the-scenes lobbying, events, letters to the editors, and any other effective means it can develop.

The Image of Computing Task Force will work to increase the public's understanding of computing, thereby increasing the number of computing students at all levels K-12, undergraduate, and graduate. The challenges are to:

- Convey positive images of career opportunities in computing.
- Counter concerns about job security created by the dot-com crash and the outsourcing scare.
- Help high school students and others know there is more to computing than AP programming.
- Help publicize the highly successful ways of introducing computing to college students that do NOT scare the students away such as those of Mark Guzdial at Georgia Tech and Randy Pausch at CMU.
- Encourage the very brightest college students to study computing.
- Encourage more CS undergrads to go on to grad school.

This is much more than a CRA-centric undertaking: we are working with CRA member societies to set up a leadership team that will move the effort forward.
Recommendation 12: Improving the image of computing is key to attracting more students to the discipline, and must be made a top priority within the Computer Science community. The CRA-led effort on Improving the Image of Computer Science (if this effort is now underway) could form the core of such an effort, but should be expanded to include IT technology councils and organizations.

We are not aware of any progress yet from this CRA project. We agree that this should be much more than a CRA effort. We also feel that any effect from image-enhancing efforts will take some time to be felt. This implies that work on this should be of the highest priority, and some deliverables should be provided as soon as possible.

One example of the kind of deliverable product that might be possible to generate quickly is the Mathematical Moments packets sent to all department chairs in September 2005. A packet consisted of 8 glossy 8-1/2 by 11 cards suitable for mounting on a bulletin board. Each card discussed an application of mathematics with an arresting picture and further references. As an interesting side note, almost every one of these applications used computing in an integral way. They included: From Arabic to Zulu, Packing It In, Reading Your Mind? Scanning the Unseen, Compressing Data, Canning Spam; see [AMS 2005].

4.4 Relationship between Industry and Academia

Industry and academia lack a comprehensive understanding of each other’s environment, opportunities, and constraints. University faculty generally believe their role is to educate students for a lifetime, not train them for a job. They view industry as wanting to tell them what to teach students. They see companies as primarily watching out for their own self-interest. Faculty often feel they do not have the time to cultivate and maintain relationships with businesses.

Industry representatives believe that students should have more exposure to applications of the concepts they have learned. They want colleges and universities to do a better job of graduating students with better critical thinking and problem solving skills, better oral and written communications skills, greater flexibility, a global orientation, experience in working on teams, and the ability to think outside the box. Some people in industry feel that the way faculty evaluates students through grades works against the ability of students to innovate, make mistakes, and take risks. They feel that academics view industry as a source of funding and products.

Clearly this lack of positive, productive relationships between academia and industry is destructive to the computing discipline. Better relationships should be encouraged. The ideal way to do this is for industry leaders to serve on an advisory committee of a computing department; one of the key contributions they can make is to provide input into the design of the computing curriculum. In addition, corporate executives can serve as guest lecturers in classes to enhance key themes. Better relationships might also facilitate companies' hiring of faculty to update skill sets of new or existing employees. When faculty members are applying for state or federal grants, industry assistance can range from letters of support to direct interactions with government officials. Often, when industry advocates for research and institutional support of public universities, legislators take notice. Companies and universities are both interested in
Recommendation 13: There are many potential opportunities for academia and industry to grow a more productive interaction in the educational arena, including increased internship, co-op, and entrepreneurial programs, two-way faculty/industry-practitioner exchange, and industry/academic collaborations and advocacy on state-level activities (particularly for public institutions of higher education). Federal agencies can encourage such interactions by providing incentive matching for innovative industry/academic collaborations, creating 3-way (industry, academia, federal) partnerships.

attracting women and minorities to university computing programs; better relationships could notably impact the quality and quantity of these efforts. Companies view universities as a key source of innovation. That relationship should be exploited in positive ways for both constituencies. Corporate executives can partner with faculty members and, where appropriate, license research that furthers their corporate goals, providing a revenue stream back to the faculty member and the academic institution. This type of university-industry partnership is an important engine for regional economic growth and should be more broadly encouraged and supported.

Better academia-industry relationships will also directly benefit students. Students should not just view the corporate world as a source of jobs; they should understand that industry is interested in being a partner in their education. It is widely believed that providing every student with at least one industry-related project that facilitates student interactions with industry professionals would be a beneficial college experience. Corporate internships and co-ops have proven to be very effective. Industry can sponsor on-site student research competitions (e.g., AT&T Research Student Research Day [AT&T 2001]) in which students present their work to industry professionals. Such events may result not only in feedback but also in start-up funding for marketable products and companies. Companies can present classes with problems they are currently facing so that teams of student "consultants" then can be empowered to research and design potential solutions.

Academia and industry can also partner with and through technology industry associations. (A list of state and regional associations may be found at http://www.crita.org). Faculty and students can attend association gatherings featuring key industry leaders and technology visionaries as well as programs on technology issues and trends. Students interested in entrepreneurship can assist in staffing investment conferences in return for access to leading venture capitalists. Many technology associations are involved in community projects that involve bringing technology into schools and community organizations. Students would be a tremendous asset in these efforts which would provide socially responsible learning opportunities (see Section 2.3).

To maintain our country's technological leadership, it is important that these two groups work together to inspire the next generation of innovators and entrepreneurs. It would be ideal if every Computer Science department were able to have students meet and interact with technology executives who have started their own companies to understand what it is like to create a product, company or market. Also important would be the ability for students to participate in such activities as technological entrepreneurship classes, business plan competitions, research projects that have the potential for licensing or technology transfer opportunities. Working together, we can ensure that the next wave of innovation has its roots here.
It is also critical that these two groups work in partnership to improve the image of the computing profession. Ensuring an adequate supply of technically skilled professionals is the biggest issue confronting the US technology industry today; recruiting more students into the academic computing pipeline is equally a challenge. It is a shared problem that will best be solved by a shared solution. If a national image campaign is undertaken (see Section 4.3), industry can be called upon to dedicate a portion of their advertising and marketing budgets to support such an effort.

The bottom line is that better relationships between academia and industry are absolutely essential. The benefits accrue to both, the upside is huge and the alternative is not acceptable.
References


[RFelder] Richard Felder (http://www.ncsu.edu/felder-public/) is an ASEE (American Society for Engineering Education) award winner who conducts workshops at ASEE conventions and elsewhere. He maintains a website RESOURCES IN SCIENCE AND ENGINEERING EDUCATION. Since 1988, he has been a regular columnist for Chemical Engineering Education.


[DPatterson1105] Dave Patterson, “Restoring the popularity of computer science”, *Communications of the ACM*, Volume 48, Number 9 (2005), Pages 25-28


Appendix A: Workshop Agenda

NSF Northeast Workshop on Integrative Computing Education and Research (ICER)

Hyatt Hotel, Financial District, One Avenue de Lafayette, Boston, MA
November 3&4 2005

Workshop Program

The planned workshop program is shown below. Because of our strong desire for an interactive workshop, with lots of discussion, the workshop has been organized into sessions with a relatively small number of talks, and ample time for open discussion.

Thursday November 3

Continental breakfast 8:45 - 9:30

Session 1: Introductory Session. 9:30 - 11:00 (Chairs: Jim Kurose, Barbara Ryder). Welcome, program overview, workshop goals and logistics, introductions. Speakers (10-12 minutes each):

- Caroline Wardle, National Science Foundation
- Jane Prey, Microsoft Research
- David Gries, Cornell University
- Lynn Stein, Olin Engineering

Break 11:00 - 11:20

Session II: Looking inwards, looking out, and integration.11:20-1:00 (Chair: Susan Merritt) Speakers (~20 minutes each), followed by moderated discussion:

- Mike Dunn, University of Indiana
- Mary Shaw, CMU

1:00 - 2:00 pm Buffet Lunch

2:00 - 4:00 pm Breakout discussion groups. Breakout into discussion groups. Informal presentation/discussions in small group setting of topics that each participant is "passionate" about. Identification of key issues, best practices, steps forward, findings and recommendations, programmatic possibilities. Each session will have a moderator, and a scribe (who will speak at the following "report-back" session). The questions below are representative; participants are encouraged to come with their own questions (and hopefully some proposed solutions and/or steps forward!)
• **Inward-Looking Discussion Group (Moderator: Barbara Ryder, Scribe: TBA).** How do we begin the process (or have we begun the process) of creating a truly foundational definition of our core on par with the core of Physics or Chemistry. What are effective "gateway" courses into the CS major? How do we do a better job at emphasizing 'conceptual knowledge' not skill building in our entry courses? How do we formulate a curriculum in which concepts learned early can be linked to specific subareas of computer science explored later? Does a "one size fits all" model of a CS major work (even among CS majors, much less a broader set of students) and if not, how do we provide flexibility? With an ever-expanding field, how do we provide a "foundation" in a relatively small number of courses (a much smaller number in 4-year liberal arts colleges than in research-intensive university departments, particularly in engineering schools). What curricular barriers are there to increasing the representation of women and minorities in computer science, and how can we most effectively attack this problem as (i) a group (ii) at our individual institutions (ii) in partnership with CSAT, (iv) with 9-12 educators? How can we educate/create "systems thinkers"? Are we providing CS students with the foundations and skills that are needed/wanted by industry? (Do we oversell or under/over emphasize the career training aspects of CS)? If we think artifact-based courses (e.g., operating systems, compilers, networking, database) as horizontal, is there a vertical approach to CS? How do we improve the image of the life of computing workers, and the computing field? What programs represent best practice, and where is innovation here occurring?

• **Outward-Looking Discussion Group (Moderator: Ursula Wolz, Scribe TBA)** In a world in which computing technology, theory and methods dominate information access and decision making, how can we reach out to other disciplines to incorporate foundational computer and information science into the undergraduate CS core, and how does CS contribute to the core in other disciplines? What relationships are possible/effective with IT programs, IT minors, IT fluency, computational science, multimedia, art and other disciplines? How can/should CS faculty best participate? Can we teach the fundamentals of computer science (not necessarily the fundamentals of programming) without training students how to program. How do we reconcile the overarching focus of a service course with the highly techniques-based approach of first year computer science courses. How can we retain the 'core' concepts and skills of computer science in our major, while shaping a curriculum in which many students taking our courses will be majoring in some other subject, yet will want to acquire deep knowledge of and skills in computer science? Does an outward looking focus provide opportunities to attract and retain students (particularly under-represented groups) to computing, albeit not in a CS major. How does CS best contribute to the broadly-educated, intellectual agile student who does not major in CS but needs fluency in selected CS topics? Do we meet industry need for CS/IT fluent students who major in a non-CS discipline? What programs represent best practice, and where is innovation here occurring?

• **Integration Discussion group (Moderator: Susan Merritt, Scribe: Jim Thomas).** How does the "looking in" lead to the "looking out"? That is, as we focus upon the creative and engaging teaching of CS to retain and excite CS students, do we build in the question of where/how this is useful to other disciplines? How do we let those other disciplines know that it is useful to them? How do we know that it is useful to them? Similarly, how does the "looking out" inform the "looking in"? For example, how does the business community's virtually universal dependence upon a tool such as Excel, inform the teaching of CS (if it does)? How does the social scientist's use of statistical tools inform the teaching of CS, if it does? Organizationally, will the CS faculty teach IT? are
they equipped to? do they want to be equipped to? Are there others on campus who are more equipped to? Do we know what IT is? Finally, how do we enable our non-computer science colleagues (both in our institutions and in our institutions’ “feeders” - high schools) to understand computer science as it relates to all that they do? if it does? and assuming that it does, demonstrating that in convincing ways? What programs represent best practice, and where is innovation here occurring?

4:00 - 4:30 pm Break

4:30 - 6:00 pm Session IV: report back (Moderator: Charles Kelemen). The scribes from each of the three breakout groups will overview the discussion in their breakout group (15 minutes each, with 10 minutes discussion following each presentation) and present findings, recommendations, etc. We'll conclude with a 10 minute "exercise," asking everyone to summarize, identify important outcomes so far (what do we want to stress in our report), identify unanswered or insufficiently addressed topics, etc. We'll use this input and the scribes' presentation to structure the Friday morning discussion/

7:00 pm Dinner at Legal Seafood. Meet at 6:45 in the hotel lobby. Self-taxi to the restaurant.

Friday November 4

Continental breakfast 8:30 - 9:00

9:00 - 10:30 Open discussion (Moderated by workshop organizing committee)

- Are there consensus findings/recommendations of the workshop? Where do we disagree?
- What steps forward can we recommend, particularly for agencies interesting in funding ICER education innovations?
- What structure do we want for our report? Who would like to volunteer to help with what sections. (Note: "many hands make for light work!)

10:30 - 11:00 Break

11:00 - 12:00. Late binding of topic. We will continue the group discussion, or divide into breakout groups.

12:00 - 1:00 Buffet lunch

1:00 - 3:00 Report-writing break-out. Break out into groups by report section. Outline sections, begin writing.