

# Outside the Box — The Changing Shape of the Computing World

Steve Cunningham  
Computer Science  
California State University Stanislaus  
Turlock, CA 95382  
rsc@csustan.edu  
<http://www.cs.csustan.edu/~rsc>

Something happened to computing while many of us were busy practicing or teaching our craft, and computing is not quite the same thing we learned. We can ignore this change and others will take our place and teach about it, but they will not have the context and the skill to understand the technology behind it and to carry it forward to the success it should have. And if others carry that torch, computer science will be stunted because we didn't recognize and respond to the opportunity.

What happened? Simply this—that Xerox and Apple and, yes, Microsoft opened up the box labeled “CAUTION: Computer Inside” and let the user into the computing picture. Users responded hesitantly but increasingly eagerly and now expect the computer to work for them instead of their working for the computer. Every application now in wide use, and any system that supports a general market, has evolved to meet that expectation. Any computing education that does not pay attention to the user's role in computing is missing the most vibrant and exciting part of computing today.

As it says at the top of this page, you are reading an editorial, not an academic paper. As an editorial, this note represents my personal passion and commitment to the user communication part of computing that the “official” computing establishment has long discounted, and I appreciate John Impagliazzo's offer of this forum to make my case that computer science is missing the boat in not understanding the need to reshape computer science education to fill this void. I would like to note that my own background has led me to shape my arguments in terms of computer science, but everything in this note applies equally well to computer information systems and only the length of the phrase “computer science and computer information systems” keeps me from using that throughout the note in place of “computer science.”

## What is the box, and what is inside and outside it?

“The Box” is the computer itself and its associated systems. It is the hardware, the operating system, the compilers, the networks, and all those things that are required to make a system work. It is, in short, the subject we traditionally study as computer science, and so we would say that the traditional computer science program is “inside the box.”

The part of computing that is “outside the box,” then, is that part of computing that does not deal with the system but with the user: the part that handles user-generated events, presents information to the user in highly understandable ways, and allows the user to work with the information in the computer in a natural and intuitive way.

In some ways, it makes sense to say that computing “outside the box” is naturally broken down into two parts. One part deals with getting information into the box from the user, and one part deals with getting information from the box to the user. I'll follow that breakdown here.

Getting information from the user into the box is the area called *interactive techniques*. This goes far beyond the simple WIMP interface (Window Interface with Mouse and Pointer) of traditional windowing systems; it includes technologies such as sensing positions and gestures in immersive

virtual environments, scanning space for user locations and motions in spatial presence environments, processing sound for verbal or musical input processing, or interpreting scanned video for personal interactive spaces. An excellent introduction to just how creative and eclectic these techniques can be is found each year at the SIGGRAPH conference's Emerging Technologies exhibition [CA98]; typically you will find all the techniques described above and more in any single year's exhibition.

Getting information from the computer/box to the user uses techniques known generally as *digital media*. The computer is capable of producing output in whatever medium best serves to communicate a particular set of information to a particular user, and it is this communication, not the media, that is important. If your user needs images, generate computer graphics; if the user needs sound, generate digital sound; if the user needs movies, generate digital video (and we are finding more and more that adding motion to computer graphics provides much more information than static graphics.) This is not "multimedia" in the buzzword sense of putting a bunch of images and animations and buttons and widgets on the screen for the user to play with; it is carefully designed and tested computer presentations that give information to the user so that the information can be used in the most productive way.

The two areas of input and output do not live independently. Effective communication with the user is the main focus of human-computer interaction (HCI) and is in the realm of ACM SIGCHI and its conferences and publications. This often requires that users be able to manipulate the output or select their own presentation techniques to get more effective information. When you have output, make the output interactive so the user can examine the information visually. When you have input, display the results of the input so the user can ensure that the proper input is given. Input and output are synergistic; used together they can accomplish more than either alone. The overall subjects of interactive techniques, digital media, and human interaction make up a unified field we will call *User Communication* and that we believe should be an important part of computer science studies.

These techniques also do not live on a single box with a single user. As we move toward the 21st century, more and more they live on the networks and connect users, information, and processing that may be far apart. The information and processing can even disappear when computers and networks are used for computer-mediated remote collaborations, one of the more interesting developments of recent years [CB97].

### **Why is it important that students know things that are outside the box?**

A real understanding of the importance of user communication leads to a number of changes in the way students need to see and understand computer systems, and leads also to a number of changes in what they need to learn in order to be fully effective in building life-long careers in the field.

Not everyone who graduates in computer science is going to write compilers or operating systems. Not everyone will work in artificial intelligence or in writing software tools. Not everyone will work in computer architecture or networking internals. Many students will find interesting and worthwhile careers developing tools or applications that support users' work with information. Even people whose careers start out inside the box may well become managers who work with projects needing high-quality user communication—graphical information output or highly-interactive control. A course of study that leads students to believe that computing is all about languages and operating systems and AI and architecture will be poorly equipped to understand the unique and critical nature of the user as part of the system.

One of the key reasons why user communication is important even to "inside the box" professionals is that the box itself must be engineered to manage the needs of this kind of work. There are differences in software and hardware design between systems that provide high-quality

user communication and those that do not. It took several years for systems to adapt to managing WIMP interfaces well, but these are much less demanding than today's and tomorrow's leading-edge communication needs; it will take several more years for systems to be able to support immersive output and tracking input, just to name two communication demands.

The reverse of the situation above is also true. It is obvious (to me, at least) that many of the persons working in the computing field but outside the box must know the kinds of fundamentals that are part of the traditional computer science program. That is, they must understand what happens inside the box and know how to work with systems in order to make the things happen that support the work outside the box. Another program (which shall go nameless) on my campus teaches Visual Basic as their answer to developing interactive systems; this isn't enough to manage the kinds of sophisticated communication that systems are going to need to provide. This is a key point in why user communication must be part of, not outside of, the field of computer science.

New computing fields career paths have developed where the key areas of work are displaying and interacting with representations of complex data. One of these is scientific visualization. At a symposium at Princeton in 1988, representatives of the National Science Foundation stated very strongly that they would not fund work in scientific visualization; only the science itself was fundable. This has begun to change, and visualization is now a fundable part of scientific research. The central nature of visualization in science means that visualization specialists are now needed in industry and university research environments, and the primary skills they must have are skills in graphics systems, data management, and interaction development. How many computer science programs offer courses in scientific visualization? The number is small but growing, and SIGGRAPH is developing curriculum recommendations in this area. But how many students have ever written interactive programs that present significant data sets visually? And how many students have programmed in tomorrow's user environments of space-sensing input and digital sound output?

An observation from one of my computer graphics classes may illustrate the value this course can add to the overall computer science curriculum. In our traditional classes we teach sorting and emphasize the difference that the right sort algorithm can make, but students often take this as a theoretical, not real, distinction. I have observed that students often choose the easiest sort to write, not the best sort, for many projects, and are not able to see the difference in algorithms when they are used on typical student datasets. But when we sort polygons on depth in a scene, it's easy for a student to see tens of thousands (or even a million or more) polygons in a scene, and suddenly they see sort times in the tens of minutes or even, if their sort was chosen poorly, in hours. They actually save time if they work harder to implement a good sort instead of implementing an easy sort, and the lesson of choosing the best algorithm is suddenly really learned.

Other aspects of algorithms are quite different for areas such as computer graphics than they are in traditional areas. Continuing with the theme of sorting, often one need not fully sort data in order to achieve the needed results. For a depth-sorted image, for example, one only needs to assure that for each pixel on the screen, the objects whose representation would use that pixel must be sorted. One thus has the notion of a partial sort, where one does only sorting that will affect overlapping areas. In many other areas, graphics provides examples of specialized algorithms or data structures that are adapted from the ones usually taught and that provide excellent examples of how one builds on and adapts classroom content to fit real-world situations.

This is a special case of finding a representation of a problem that has a superior computational approach while maintaining the user's understanding of the information. Graphics has long dealt with producing convincing images while keeping computational complexity as small as possible. This is part of the more general problem of providing all the information a user needs without carrying out computations that do not add to the value of the output to the user. This requires a

deep understanding of the processes of human vision and of the way a person chooses among options and manipulates devices—long staples of the computer graphics and HCI worlds but certainly not part of traditional computing studies. As we consider the challenge of extracting useful information from vast amounts of data we find the same kinds of tasks, and students who have met the problem in a graphical environment can understand the issue better when it arises in a different domain.

### **What should a student know about things that are outside the box?**

As I note below, it is a challenge to define the user communication part of a computer science program. Instead of taking on this task, I want to suggest some aspects of this area that can be of value to the general computer science student. There are fascinating pieces to these technologies whose understanding and mastery would be extremely valuable to many of our students, and with which all our students should have some familiarity. Others will extend this list further, but we might as well start.

One of the important facets of interactive technologies is the notion of event-driven processing, where students must think in ways that parallel—or even use—the operating system process-dispatch level services. This might take the form of threaded languages or of dealing with the event queue, but in any case it provides an excellent example of managing system resources.

These user communication technologies involve standards, often developing and sometimes competing standards, and studying these standards would help students understand the role of standards, follow the standards development process, and learn how to evaluate competing standards. This area also offers the opportunity for advanced students to work with areas for which there are no standards yet and for which the student may have to deal with questions whose answers are not fully known.

Students need to understand that information is different from data, and that there are many ways to manipulate data to achieve information. Many of those ways are highly interactive and allow the user to define the information she needs while the processing is happening; others require a study of information properties and a careful development of information presentation in intuitive ways. This begins to get away from computing and move toward issues in HCI, and these need to be part of user communication studies. However, the full discipline of understanding human work and communication processes that is part of HCI is probably not part of undergraduate computer science unless user communication is interpreted much more broadly than even I am suggesting.

ACM SIGCHI defines HCI as “a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.” The content of HCI [CH92] includes computer system and interface architecture, comprising I/O devices, computer graphics, and dialog techniques, genre, and architecture. It also includes the development process, comprising design, implementation, and evaluation techniques. The SIGCHI study of curriculum includes several course outlines, of which “User Interface Design and Development” could fit comfortably alongside most software engineering courses in computer science. However, it is important to note that *designing* interaction is a very different discipline from any traditional software process. The design process deals with humans and how they work, and software processes in this area deal with creating the capability to support interaction styles and implementing particular interaction design.

Finally, not every possible course that uses interactive or graphical tools is a course for the user communication area. We must differentiate between the technology of the user communication areas and the tools that serve the end user. Computer science should focus on the technology of user communication, not on the commercial tools that are used in the area. Computer science programs’ role in teaching end-user tool use belongs in service courses, not in professional

courses. It does not make sense for computer science to teach an advanced course in PhotoShop when there is a photography program or a graphic arts program that could teach it in the context where it is used.

### **Why aren't current curriculum models adequate to address this question?**

Curriculum models have the nature of informal standards, and in fact the reliance of accreditation on these models almost make them formal standards. Thus it can be interesting and enlightening to look at them in the light of standards. As I mentioned above, there are many standards for digital media, and studying them leads to a study of standards processes. Quite often standards are designed to unify processes that are already in place but that differ in awkward ways from place to place; this seems to be the nature of curriculum standards. Early computer graphics standards such as GKS were certainly of this nature. But in unifying existing processes, standards may have the unwanted effect of freezing the state of current practice. When they do this in any dynamic area, they quickly become outdated and lose their value. And computing is among the most dynamic and changing areas in human history.

So we should ask the question of whether our current curriculum models have stood up well as standards. It is clear that many parts of them have. We do need to understand the theory of computation, we do need to understand the social and ethical impact of computing, we do need to understand computer organization and architecture, and the list goes on. But in other ways the curriculum models have served to stunt the growth of computer science education by focusing entirely on what happens inside the box. It does not take much observation of the curriculum models to see this.

For example, Computing Curriculum 91 [CC91] does not represent “outside the box” material at all well. It suggests that a very small amount of computer graphics and user interaction be taught in an overall undergraduate program. The Human-Computer Communication portion of the Computer Science core comprises eight lecture hours that are to cover both user interfaces and computer graphics. Nothing is included that suggests the value of anything beyond WIMP interfaces; nothing is included that suggests the value of any other kind of output besides still images; nothing is included that suggests any real depth to any of these topics.

The [CC91] treatment of human-computer communication does suggest that the computer science curriculum should include work beyond the core, and among a list of 28 possible areas are courses in computer graphics and human-computer interfaces. Similarly the Computer Science Accreditation Board criteria [CA96] suggest that an advanced course in human-computer communication (one item in a list of 11) is an appropriate part of a computer science program. But both are presented in such a way as to minimize the perceived value of such courses, especially for the accreditation process where most of the emphasis is on a rather rigid list of core areas that are strictly inside the box.

The list of electives in [CC91] includes samples of possible courses. There is a sample course for computer graphics; there is no sample listed for user interaction, for input technologies, or for any other kind of digital media. These are important oversights. But the sample course also have the nature of standards since they offer suggestions on which to build courses, and these have themselves become outdated. There have been significant changes in computer graphics since the computer graphics sample course [CC91] (p. 72) was laid out, which I can say because I wrote that description. That course followed the traditional “fundamental graphics concepts” elective course described in [BR88] as one of several kinds of graphics courses. However, I believe that the first course in Computer Graphics should now be fundamentally changed. It would now emphasize modeling and spatial thinking, would use a modern API such as OpenGL, and would de-emphasize pixel-level techniques. A possible such course is described in [CU98] and reflects the increasing number of introductory courses being taught based on OpenGL and other current

APIs. However, other topics and other courses need to be developed to fill out the user communications area.

Interestingly enough, the new model curriculum for computing in liberal arts colleges [WS96] is no better than Curriculum 91. It may even be worse, because the recommended eight lecture hours on human-computer communication are reduced to three in the liberal-arts curriculum. The only hint in that curriculum that a human may be involved with the computer comes in the electives, where graphics is among 17 possible elective courses; at no point is there any indication that overall human-computer interaction or an expanded view of computer output is recommended for computer science. And this is in the liberal arts, where one once would have assumed that human issues would be taken more seriously than in large research universities.

The one voice on curriculum recognizing the need to include new material such as user communication is Eric Roberts in [RO96]. His recommendation 4, *Some classical topics must be deemphasized to make room for the new material*, specifically mentions HCI and graphics as critical topics that must be included. User communication is simply a unifying concept for these two areas as well as more technique-oriented concepts in user input and richer output.

### **How can we expand computer science education to include these areas?**

It is challenging to ask ourselves how we can teach the kinds of user communication—input, output, shaping information—that we have been discussing above. The challenges include creating overall curriculum components, designing courses or other study units for these components, building laboratories that support these courses, designing laboratory work to be part of the courses, learning enough in these content areas to make it possible to teach solid courses, and developing texts and teaching materials to support these courses. We need to do this both for a single course that could be appropriate for any computer science student, and for a focused program that could become a concentration with a computer science major. This is a daunting project.

The first piece of the project must be to understand the kind of components the curriculum needs to have in user communication. What kinds of input should a student understand? What kinds of output? What application areas will provide the right kinds of problems that will motivate and reinforce student learning? What general communication principles underly user communications in computing? I would suggest that scientific visualization would provide a valid framework on which to build these components, but other application frameworks could work as well or better.

Once the framework and components of user communication are understood, we need to develop lab experiences that will support the student learning of the material. The nature of user communication studies is likely to require working in a very tool-rich environment, where students can work with scanned and sensed input as well as a variety of input devices, and can work with many kinds of graphical and other output. Building such laboratories will also be very challenging, particularly given the budget constraints most of us work under.

The components and laboratories then need to be organized into courses in user communication that fit into the overall curriculum of the computer science program. Because this subject matter is outside the recommended curriculum areas, it may be challenging to get faculty or administrative approval for such courses, and it may be difficult to get accrediting bodies to understand the allocation of major resources to them.

Once you have the courses, you need skilled faculty to teach them, and most of us do not have backgrounds in user communication. Faculty development will be needed, and there are few places outside major research laboratories where faculty can now get the background to teach this material skillfully. You also need good teaching material for these faculty. However, there are

very few textbooks in the areas of sophisticated computer input and digital media output. Most of the books and teaching materials are actually training materials for media applications, and so are not appropriate for the kinds of studies we are suggesting. Textbooks will have to be written with general concepts in mind.

With so much to be done, and with so little history of undergraduate teaching in this area, teaching faculty are going to need a great deal of help in creating the area of user communication. We are going to need to develop collaborations with research laboratories and professional societies in the field, and we must expect that it will take several years to develop the first courses in the area. Fortunately we have the right kinds of groups within ACM to undertake this work. If the SIGs in this area—SIGGRAPH, SIGCHI, SIGMM, and SIGCSE—will work together, we can create this new area and keep both ACM and computer science education on the leading edge of computing developments.

### **Acknowledgments**

Learning what the box is and how to think outside it has been a long and interesting process, shaped by my many friends and colleagues in SIGGRAPH. I cannot begin to thank all of them, but I would like to thank Judy Brown, my co-author on two books and many articles who has shown me how much people can accomplish through visual thinking, and Maxine Brown, who opened my door to SIGGRAPH in the first place and whose creativity has long challenged me. And finally I would acknowledge my colleagues at CSU Stanislaus who have supported my own work outside the box.

### **References**

- [BR88] Brown, Judith R. et al., “Varieties of Computer Graphics Courses in Computer Science,” Proceedings of SIGCSE 88, 1988, *SIGCSE Bulletin* 20(1), 313
- [CA96] Criteria for Accrediting Programs in Computer Science in the United States, Computer Science Accreditation Board, 1996
- [CA98] SIGGRAPH 98 *Conference Abstracts and Application*, *Computer Graphics Annual Conference Series*, ACM SIGGRAPH, 1998
- [CB97] Cunningham, Steve and Judith R. Brown, “Computer Graphics: From ‘Interactive’ to ‘Collaborative’,” *ACM Interactions*, 4(4), July/August 1997, 88-87
- [CC91] Tucker, Allen B. and Bruce H. Barnes, eds., *Computing Curricula 1991: Report of the ACM/IEEE/CS Curriculum Task Force*, ACM Press/IEEE Computer Society Press, 1991
- [CH92] ACM SIGCHI Curricula for Human-Computer Interaction, ACM, 1992
- [CU98] Cunningham, Steve, “An Evolving Approach to Computer Graphics Courses in Computer Science,” *Proceedings of GraphiCon 98*, Moscow, September 1998. Online at <http://www.cs.csustan.edu/~rsc/GC98.pdf>
- [RO96] Roberts, Eric, “Directions in Computer Science Education,” *ACM Computing Surveys*, 28(4es), December 1996
- [WS96] Walker, Henry M. and G. Michael Schneider, “A Revised Model Curriculum for a Liberal Arts Degree in Computer Science,” *Communications of the ACM*, 39(12), December 1996, pp. 85-95