Threads™: How to restructure a computer science curriculum for a flat world

Merrick Furst, Charles Isbell, and Mark Guzdial
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, Georgia, 30332-0280
{furst,isbell,guzdial}@cc.gatech.edu

ABSTRACT

In his book *The World is Flat*, Thomas Friedman convincingly explains the challenges of a global marketplace [4]. One implication is that software development can be outsourced, as can any narrow, skills-based occupation; however, as Friedman also points out, leadership, innovation, and insight are always in demand. We have recently created and are implementing *Threads*^{TM}, a new structuring principle for computing curricula. Threads provides one clear path for computer scientists seeking to reinvent and re-invigorate computer science degree programs. Threads form a cohesive, coordinated set of *contexts* for understanding computing. The union of all threads covers the breadth "computer science." The union of *any* ***two*** *threads* is sufficient to cover a computer science degree. In this paper, we describe Threads, our process, the impact so far, and some of our future plans. We close with recommendations for other schools, especially schools with smaller programs.

Categories and Subject Descriptors

K.3 [**Computers and Education**]: Computer and Information Sciences Education

General Terms

Experimentation,Design

Keywords

curricula, programming, off-shoring, out-sourcing, flat world

1. INTRODUCTION: THE CHALLENGE OF A FLAT WORLD

In Thomas Friedman's best-selling book *The World is Flat*, he highlights the problems facing computer science faculty in the United States [4]. In a flat world, workers anywhere can compete on a level playing field with workers in the United States. Jobs such as accounting and even software development can be performed as well, or even better, off-shore as they can within the United States. Any job that can be defined as a narrow set of skills—no matter how intellectually challenging—is completely exportable, and often at an apparent lower cost than the same job can be performed in the United States. Friedman's focus was on the United States but his analysis is generally valid.

Friedman also suggests the nature of a solution. The jobs that can be sent off-shore are those that are *decontextualized*, which means that there are few connections between the job and others, between the discipline of the job and other disciplines. To be globally competitive, students need something beyond a narrow set of skills. They need a broad computing education so that they may become experts at mastering context and working in multidisciplinary settings.

Friedman says that "The job of the U.S. is to innovate." That statement is a historical fact, not a promise for the future. Others can be just as smart or smarter. We recognize that we must come up with new kinds of innovation, and that our students need those meta-innovation skills to be competitive.

The lesson for educators is the same as for the students. We must innovate in our curriculum, in how we approach what we teach students. The need for us to innovate is driven both by the need for us to be role models for our students and by the challenges faced by our field. Innovation is the way to compete for students, and to re-establish and reinforce the relevance and importance of our discipline.

1.1 Our Approach: Context and Motivation

Daniel Pink argues that we are in a move from an Intellectual Age to a Conceptual Age [9], driven by some of the same forces identified by Friedman. Pink identifies several important trends and new skills that are important in this shift of emphases. In particular, he points out that those who innovate will demonstrate *symphonic thinking*, the ability to integrate ideas across different disciplines or fields. Integrating across several fields facilitates insight and innovation. We found a resonance between Friedman's call for context and Pink's notion of symphonic thinking.

Students need to understand *why* they are studying what they are studying. This is important for innovating, integrating, and facilitating the most basic learning. Learning happens when we connect new information to existing knowledge [7], and unless we activate that prior knowledge

SIGCSE'07, March 7–10, 2007, Covington, Kentucky, USA.

first, students only have "brittle knowledge" [3]—knowledge that will get them by tests, but will not be successfully used outside of class. By making students aware of the context that makes what they are studying important, we have an opportunity to both improve learning and dramatically improve retention [2]. Studies of higher education show that retention is improved by simple measures such as improved counseling that explains to students why they are taking the classes that they are taking.

The need for context may also explain some of the decline in student interest in computer science. Why do students enter computer science? Most students who apply for the CS major, at least here at Georgia Tech, have not had high school computer science. They express interest in computer science because they are interested in video games, or perhaps the technology behind cell phones and digital audio players, or maybe with the dramatic impact of computing in areas like the Human Genome Project; however, historically when they enter their first computer science class, they hear nothing about any of that, instead focusing on sorting algorithms and traveling salesmen. The context that the students are interested in, that they came for, does not match with the experience. It is no surprise that the students find this unattractive.

Teaching traditional computer science as we have, we might imagine that we are teaching a context—the context of computer science as we know it. We may define our job as expressing that traditional computer science context to students. But that is a less than effective model, as research on motivation has shown us [10]. When students feel as if they have a choice in what they do or what they study, they are more motivated, and motivation plays a significant role in learning [1].

2. THREADS™

Threads are new cohesive, coordinated contexts for understanding computer science [5]. The union of the threads at a given school or department provides a local definition of computer science. The union of the eight threads that we use at Georgia Tech's College of Computing currently defines computer science for us.

A thread serves as a context for interpreting the courses in a curriculum for both students and faculty. A thread makes its set of courses cohesive, providing an overall meaning for the set and individual meaning for each course. A thread also suggests a coordinated path through its courses so that the end result is expertise in the area of the thread. A thread is therefore a *trajectory* that leads through a set of courses, drawing them together towards an particular end.

We currently define eight threads:

- **Foundations:** Fundamentals of computing, such as computer science theory.
- **Embodiment:** Computing meets the physical world, in such areas as robotics and real-time embedded systems.
- **Intelligence:** Computing as cognition, its representation and processes.
- **Computational Modeling:** Computing for representing the world, as in computational sciences.
- **Platforms:** Computing across different kinds of hardware, with different characteristics and infrastructures.
- **Information Internetworking:** Computing for storing, recalling, and communicating information.
- **People:** Computing meets people, including the design of human-centered systems.
- **Media:** Computing for processing, creating, and presenting multimedia.

Many traditional computer science courses fall naturally into each of these threads. For example, a course in human-computer interaction naturally falls under the People thread, and a course on robotics naturally falls under Embodiment. On the other hand, as we shall see later, it is a mistake to think of each thread as a traditional research area or specialization. It is a mistake to think of People as a fancier name for HCI or for Embodiment as a fancier name for Robotics.

Threads include courses outside of computer science. Given the freedom to think of a broad range of disciples that complement computer science, some psychology courses become obvious candidates for inclusion under People and Intelligence, while some courses in physics make sense for both Embodiment and Computational Modeling.

Introductory courses are important, just as they are for traditional computer science curricula, but not all courses are important for all threads. For example, our introductory courses in software engineering do not necessarily make sense for the People or Media threads. Different threads have different introductory course requirements, *e.g.*, the Media thread draws upon our media computation classes [6]. Some courses are useful for many threads, but for reasons that differ depending on the thread in question. An introductory course in systems, for example, informs students in Platforms about different kinds of infrastructure, and informs students in Embodiment about support for real-time programming.

If you asked 50 computer scientists what every CS student *must* take, you would get far more than 50 answers, and probably end up with a five year (or more) degree program. We refer to this as *the core problem*. It is a powerful impediment to a establishing a flexible degree program. Threads does not guarantee that every core course or topic is included. Rather, it guarantees that each thread is coherent, that courses make sense within that trajectory.

2.1 Threads and Degrees

A BS in Computer Science at Georgia Tech is now defined as *any two threads*. While any one thread only contains some of what might be called "core" topics, every thread *combination* has enough overlap with the "core" to be called a computer science degree by nearly any definition. Every student in Threads still takes some data structures, some systems programming, and some software engineering, as examples. Each combination still has all the necessary pieces that accreditation organizations like ABET recognize as a computer science degree.

In practice, there is significant overlap between threads, *i.e.*, the same course can make sense in two (or more) threads, but with possibly different reasons for taking the same course within different threads. Any two threads still leaves credit hours for students to use on elective courses. The result of

the *any two threads* rule means that there are now $\binom{8}{2} = 28$ different combinations that lead to a degree.

In our process for defining Threads, we separated the breadth requirements of the Institute. Requirements for social sciences, humanities, and laboratory sciences, for example, were left separate from the threads. Thus, all combinations already meet all the requirements of the Institute for an undergraduate degree; however, some threads do make demands of those breadth requirements. For example, both Embodiment and People require introductory psychology to be taken as one of the social sciences. At Georgia Tech, this was the general process by which requirements were "added" to each Thread without increasing the number of hours or reducing the amount of computer science depth in the program.

2.2 Threads and Careers

The relation between threads combinations and real world careers is many-to-many. Most real world contexts for use of skills and knowledge in which students might be interested actually map to two or more combinations, depending on the aspect of the context most interesting. Any combination could actually be used as a stepping stone to many different careers.

For example, consider the context of *information security*. A student interested in information security would almost certainly want to study Information Internetworking as one of her two threads. But her second thread depends on interest. If she were interested in the encoding algorithms that allow information to be transmitted and stored securely, then the second thread might be Foundations. If she were interested in how secure systems are implemented in the real world across a wide variety of systems (*e.g.*, secure distributed databases), then Platforms would make a good second thread. If she were interested in building secure systems that users would be willing and able to use correctly then she might focus on People.

In the other direction, any given thread combination can lead to several careers. Consider the combination of Embodiment and People. One possible context or career focus with that combination would be *social robotics*, the development of robots who work in human social contexts and who interact with humans as agents. Another context or career focus might be *advanced prosthetics*, the development of devices with embedded computing that help to replace lost human limbs.

In other words, students can think in terms of careers they are interested in—such as information security–and find threads that support that interest, or they can think of the kinds of threads that they find interesting and discover careers that are consistent with that interest.

3. DEVELOPING THREADS

The first author, Furst, proposed the idea of Threads and co-ordinated their development. As the associate dean responsible for academic programs, he started the process of engaging the faculty in defining Threads for the College of Computing.

The second author, Isbell, led the faculty task force that defined the specific threads and the learning objectives for each. Very explicitly, they did not tackle *the core problem*. Rather, they sought to identify a set of threads that:

- Made sense given the emphases and strengths of the College of Computing.
- Made sense to our students, and the students that we hope to attract.
- Made sense to those who would receive our graduates: employers and graduate schools.

Further, in designing each thread we had several explicit desiderata:

- The number of threads should be small.
- Each thread should be internally consistent with some theme.
- Each theme should cross more than one subarea of computer science.
- Each thread should specify courses from outside of computer science.
- Each thread should contain roughly the same requirements of about 2/3 of a full degree.
- Any pair of threads should not exceed the requirements of full degree.
- The entire set of threads should cover all of what we consider computer science.

Isbell's task force explicitly engaged students and industrial partners in what they thought was important. They sought to answer questions like, "If we were to say that this student is an expert in *Computing and Intelligence*, what would you expect them to know?" Isbell's group generated a set of threads with both learning outcomes and performance goals (what should students who study this thread be able to do?).

The third author, Guzdial, led the task force that implemented threads as a degree within the requirements of our Institute and ABET accreditation. Guzdial took the output of Isbell's committee and specified the threads in terms of course requirements and electives, including courses left out of the discussion but necessary to meet requirements (such as social science, humanities, and ethics courses). This was the proposal that was given to the task force as a starting point. Virtually none of the original proposal's threads definition remained completely intact (there were even different threads that emerged from the implementation process), but the result remained consistent with the desiderata above and met all necessary degree requirements.

Some of the details of this last process are worth considering. Each thread is divided into two sections: required courses and technical electives (see Table 1 for a sample of two threads). As we've noted before, required courses sometimes includes non-CS courses. Further, some requirements are defined as *picks*: students are told to "pick m courses from a set of n courses. Typically, these courses represent breadth in the thread. For example, in the People thread a student is told to pick 2 courses from a set of introductory courses in Cognitive Science, HCI and Educational Technology. The explanation to the students is simple: at Georgia Tech we teach expertise in three particular subareas of CS and in order to be an expert in People, the student should

People

Social Science elective: PSYC1101

CS1301 Introduction to Computing and Programming
CS1331 Introduction to Object-oriented Programming
CS2340 Objects and Design
PSYC2015 Research Methods and Practices

Pick 1 of the following:
PSYC2210 Social Psychology
PSYC2760 Psychology of Human Language
PSYC3040 Sensation and Perception

Pick 2 of the following:
CS3790 Introduction to Cognitive Science
CS3750 Human-Computer Interface Design and Evaluation
CS4660 Introduction to Educational Technology

Embodiment

Social Science elective: PSYC1101
Lab Sciences: PHYS I and PHYS II

CS1050 Understanding and Constructing Proofs
CS1301 Introduction to Computing and Programming
CS1331 Introduction to Object-oriented Programming
CS1332 Data structures and algorithms
CS2110 Computing organization and programming
CS2200 Computer systems and networks
CS3251 Computer Networking 1

Pick 1 of the following:
CS3510 Design and analysis of algorithms
CS3240 Languages and Computation

Pick 1 of the following:
CS3630 Introduction to Perception and Robotics
CS4605 Mobile and Ubiquitous Computing

CS3650 The Art of Building Intelligent Appliances
PSYC3040 Sensation and Perception

Table 1: Requirements for the People and Embodiment threads. Note that each thread specifies non-CS courses. In addition to the required courses shown here, each thread also defines a number of courses that can fulfill technical electives.

have broad knowledge in at least two of those subareas. Notice that such breadth course are always available as technical electives as well, and that follow-on courses are typically listed as technical electives. Thus, the student can continue in breadth and still receive credit, or can continue in depth.

Even this last stage avoided the core problem. The implementation task force defined "Threads shepherds" whose job it was to define a single thread and oversee interactions in combinations with other threads. The interactions in combinations were hard to predict and occasionally hard to resolve, *e.g.*, when a combination ends up requiring too many credit hours and *someone's* thread has to shrink. But at no point did we ask the question, "What should a computer science student know?" Rather we asked eight separate questions, "What should students in *this thread* know?"

Relatively few new courses were needed to create Threads. Approximately 20 new courses were defined in the process, but only six of them were actually required in any thread. For example, a systems course was defined for Media and People threads, which teaches low-level hardware and systems issues in the context of programming a media device (*e.g.*, a Nintendo Gameboy), but that course had already been planned for our BS in Computational Media, so it was not an additional cost. The Embodiment thread defined a new capstone course, *Prototyping Intelligent Devices*, which is to be held in a shop class where students will actually work with metal and plastic to create chassis and bodies and embed in them their computational devices. Many of the new courses were technical electives, and in some cases undergraduate versions of early graduate courses. One side effect of Threads was that it allowed us to prepare the focused and interested students to take some of these advanced courses.

The proposal to change the BS in CS degree to Threads was approved unanimously at the College level in December 2005 and at the Institute level in January 2006. The rest of the Georgia Institute of Technology is enthusiastic about and intrigued by the potential for Threads. Other science and engineering disciplines are also facing downturns in enrollment and are also challenged by a flat world. There are discussions across campus about how to use Threads to redefine other degree programs. A thread-ed campus would have several benefits. For example, a joint degree might be formed by simply combining one thread from each unit.

We are currently supporting Threads for the incoming class of Fall 2006. That means that existing students can switch to the Threads definition of the BS in CS, but we only promise to have new courses created in time for the Fall 2006 first years. Because there are only six completely new required courses, no new courses may be necessary for existing students to graduate under Threads. But promising Threads only for Fall 2006 allows us to space out the resource requirements for creating new courses. We only need the new courses when the Fall 2006 students arrive at that point in the program.

4. IMPACTS

Evaluating a "structuring principle" is challenging, and virtually impossible in the few months since its creation. We can offer some pieces of evidence about the impact.

The response from across campus, from the College's advi-

sory board, and across the campus has been overwhelmingly positive. Industry recognizes the need for "Versatilists" [4], people who combine narrow technological skills with broader context [8]. Across campus, Threads seem to answer important questions raised by a flat world.

Thomas Friedman's revised and expanded edition of *The World is Flat* [4] has a chapter highlighting the changes at Georgia Tech, especially in the College of Computing. Friedman says that Threads is the kind of answer he thinks is necessary for keeping our students and our economy competitive in a flat world. He has praise for efforts across campus to generate integrative and innovative thinking, such as the emphasis on musical interest in admissions in order to create more well-rounded and imaginative engineers.

In Spring 2006, when admissions came in for the College of Computing, we were disappointed to learn that we had 20% fewer applications than the year before. But those students had not heard about Threads yet. We made a significant effort to contact prospective students, including those undecided students across campus, to tell them about Threads and the changes going on in the College of Computing. Of course, there was a threat that students who were applying to computer science *preferred* the non-Threads way of doing things. But parents and students seemed excited about our approach. In the end, the Fall 2006 class in the College of Computing is 20% *larger* than the year before.

5. ROLES AND THREADSPACE

We are engaged in another step in our curricular revision. Isbell's original task force defined both Threads, describing what a student should *know*, and *Roles*, describing what a student might want to *do* with that knowledge. In any Threads combination, a student might be a *Master Practitioner* (a super software engineer), or perhaps a *Researcher*, or perhaps an *Entrepeneur*, *Communicator* or *Policy maker*. We are defining activities and suggestions for students to prepare themselves for the Roles that they wish to play within their chosen Threads.

Isbell is also leading an effort to define a software space, *Threadspace* for guiding students. Threads creates some new challenges for advising. The cohesiveness of each individual thread is unhelpful, if the students do not perceive the connection between the courses and the students' selected threads. Threadspace is meant to communicate to students the meaning behind the threads, help them to choose threads and announce thread intentions, and to help teachers, advisors, and administrators to implement threads. For example, we would like to be able to tell a teacher in an introductory course what percentage of her class is interested in each thread, so that examples and homework might be tuned to reflect the contexts of students' interests.

6. THREADS FOR OTHERS

We recommend that all computing departments consider a Threads-like restructuring. In a flat world, business-as-usual is a dead-end. We must innovate in order to to lead our students and to keep our discipline competitive.

We recognize that Threads is particularly forbidding to small schools. Our recommendation to smaller programs is *not* to adopt our threads, or even a subset of our threads. Each department should play to its strengths and the strengths of its faculty. We believe that the desiderata that we have outlined here provide good guidance for developing threads, but that the particular set of threads that we have defined are not the only ones that meet those criteria. Create a local definition of computer science for your students in terms of what you can best offer and use *picks* as a mechanism for teaching specializations and breadth.

Our key recommendation is that teaching traditional, decontextualized, abstract computer science that can be applied to any context is an enormous mistake. Learning science has shown that learning about *some* context transfers to other contexts, while learning only *abstractions* tends to lead to brittle knowledge that gets used nowhere [3]. Our students cannot compete globally if we only teach them decontextualized skills that can be learned just as well and executed more cheaply on the other side of the globe. Threads gives students the cohesive, coordinated contexts that explain why they are taking the courses that they are taking and for what kinds of careers they are preparing. Our students need that symphonic view of computer science to be prepared for the flat world.

7. ACKNOWLEDGMENTS

We wish to thank Dean Richard DeMillo for his support of the Threads process and to President Wayne Clough for his encouragement to explore Threads across campus.

8. REFERENCES

[1] P. C. Blumenfeld, E. Soloway, R. W. Marx, J. S. Krajcik, M. Guzdial, and A. Palincsar. Motivating project-based learning: Sustaining the doing, supporting the learning. *Educational Psychologist*, 26(3 & 4):369–398, 1991.

[2] J. Donald. *Improving the Environment for Learning: Academic Leaders Talk About What Works*. Jossey-Bass, 1997.

[3] M. S. Donovan, J. D. Bransford, and J. W. Pellegrino. *How People Learn: Bridging Research and Practice*. National Academy Press, Washington DC, 1999.

[4] T. L. Friedman. *The World is Flat: A brief history of the twenty-first century*. Farrar, Straus, and Giroux, first updated and expanded edition edition, 2006.

[5] M. Furst and R. A. DeMillo. Creating symphonic-thinking computer science graduates for an increasingly competitive global environment. *http://www.cc.gatech.edu/images/pdfs/threads_whitepaper.pdf*, 2006.

[6] M. Guzdial. A media computation course for non-majors. In *Proceedings of the Innovation and Technology in Computer Science Education (ITiCSE) 2003 Conference*, pages In–Press, New York, 2003. ACM, ACM.

[7] J. Kolodner. *Case Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[8] E. Montalbano. Gates worried over decline in us computer scientists. *Computer World*, July 18 2005.

[9] D. H. Pink. *A Whole New Mind: Moving from the Information Age to the Conceptual Age*. Riverhead Hardcover, 2005.

[10] P. R. Pintrich and D. H. Schunk. *Motivation in Education: Theory, Research, and Applications*. Prentice-Hall, 1996.