CPATH CDEF: Resparking Innovation in Computing Education

PI: Peter J. Denning (1/16/07)

Recovering Basic Teaching Qualities

The Computer Science Teachers Association (CSTA) asked me to comment to their membership about how K12 teachers can prepare students for the skills and personal traits needed in CS careers. Here is what I said to them. It sets the stage for the plan proposed here.

THE JOY OF COMPUTING

We can help our students find the joy in computing if we cultivate in ourselves as teachers the two qualities that we admired most in our own teachers: helping us to play in big games, and providing adult leadership.

Big Games

My high school math teacher, Ralph M, was also the faculty advisor for the science club. He encouraged everyone to join the club; and he insisted that every club member prepare a project for the annual science fair. Noting my curiosity about computers, he told me to build a computer. Flabbergasted, I asked, "How can I possibly do that? There are no computer courses to teach me what to do." He responded, "Computers are going to be a very big game. In a decade or so computers will make your trusty slide rule obsolete. You can be one of the pioneers who make that happen. Go learn what you can about computers now, and build one." And so I did. My project won first prize in the fair.

In graduate school, MIT's Project MAC brought me into an even bigger game. The "computer utility" was an early view of the Internet that envisioned a universal, cheap network computing service. To access it, you would plug a terminal into a wall socket, as with the electric utility. I was part of a community developing new concepts to make the computer utility real -- concurrent process control, time-sharing, virtual memory, interprocess communication, user interfaces, object programming, packet networks, and much more. By making these technologies work, we would enrich many lives.

These two games brought me great joy. They also spread great joy through a whole generation of computer scientists.

Soon thereafter, late 1960s, the ACM Curriculum Committee sought to capture the core principles we learned so that our students would learn them faster and experience the same joy we experienced; so we thought.

But now, forty years later, the technologies based on those principles (processes, virtual memory, networks, etc) are mature. They are part of every network and every operating system. We hardly notice them at all. And here's the punch line: <u>They no longer bring the joy of discovery to those who learn</u>

them. In its ability to motivate students, the 1960s framework for understanding our core principles is obsolete.

The true sources of the joy of discovery were the applications we enabled. People in many fields could do wondrous things they could never do before.

Today, there are many more computer applications than in the 1960s. They are in art, music, entertainment, games, information management and discovery in the Web, biotechnology, genetic medicine, genetic engineering, development of new materials, nanotechnology, cosmology, deep space exploration, and more. Much joy awaits students who focus on these applications. Students have much to crow about in their science fair projects.

If our students are busy with science fair projects, when will they learn the basic principles? I say: get students to play in big games involving computing. They will taste the joy of discovery. They will become interested in the principles. More important, they will discover new ones.

Adult Leadership

Ralph M was a superb science coach. He inspired us with possible new worlds we could create through our minds. He got us to start building those worlds with our own hands. He promoted us with other people and showed us they would take our ideas seriously. He instilled in us the rigorous habits of thought needed for math, science, and engineering.

He was also a great life coach. He counseled us about our relationships, about cars and sports, about art and music. We brought all sorts of issues from our lives and he rewarded us with insights on dealing with them. He showed us how to manage ourselves. Because of his grounding in life, we could see that our own lives as mathematicians, engineers, or scientists could be rich and rewarding.

We often wonder why so many high school kids are attracted to sports rather than science and engineering. I think it's not the lure of money or professional sports; it's simply that most sports coaches are great life coaches. The students look up to them and admire them. Their life coaches are better than our life coaches.

As teachers, let us be not only science coaches, but also life coaches for our students. Our students can come to see that practicing in computing is a source of wisdom as well as joy.

Perhaps even more than adults, kids make their choices from their hearts more than from their heads. The logic of choosing a profession because of money or getting in on the ground floor is less powerful than the admiration of a teacher, hero, innovator, or leader. Playing in big games with help from adult leaders are sure winners of hearts.

Implementation Plan Overview

The Mess

We find ourselves in the midst of an intransigent mess.

US competitiveness is threatened because of a shrinking education pipeline for S&T people in the face of growing job market demand for these people. The computing field has been hit especially hard: since 2001, CS enrollments dropped by half. Current figures suggest that the computing pipeline is producing about 60% of the labor market need.[1] The drop is paradoxical since industry demand for computing professionals is projected to be a strong growth market for at least the next decade. The ACM Education Board has been trying to make sense of the paradox. Eric Roberts compiled a list of reasons that have been cited in various surveys about why this has happened:

1. Myths about employment opportunities dwindling
2. The abysmal state of high-school computer science
3. A general lack of understanding about the discipline and its subfields
4. Negative images of work (and workers) in the computing field
5. Student worldviews in which computer science is unappealing
6. Much more difficult introductory courses
7. Lack of breadth and flexibility in the computing curriculum

Roberts concludes that all the reasons appear to be working in concert. The situation is complex and there is no single solution.

Diagnosis

I have been studying innovation intently for several years, with an eye toward teaching innovation as a skill.[2] One of our findings is that messes signal that the current paradigm (belief system) is unable to cope with current circumstance or explain current anomalies.[3] Therefore something must disrupt the current paradigm before the mess will be resolved.

I believe that our framework (paradigm) for teaching computing has become outdated. People who experience our field for the first time through our traditional framework do not experience the same joy of discovery, connection to Big Games, and innovative applications, as did those who designed the framework. Too many prospective students, viewing computing through the lens of our curriculum framework, do not see a field attractive enough to merit

---

[1] BLS statistics for 2005 show 106,000 openings in CS and rising. US Dept of Education completion survey statistics 2004 show BS CS production 61,000.

[2] In 2002, I organized the Innovation Project in the Cebrowski Institute for Innovation and Information Superiority at the Naval Postgraduate School, Monterey, California. Bob Dunham, President of Enterprise Performance, Inc., is my main partner. Our fundamental hypothesis is that innovation is a skill and skilled innovators have much higher success rates than the prevailing business rate of 4%. We have identified seven foundational practices as the core of the skill. We can teach someone who engages with these practices to be a competent innovator in a year or less. We are writing a book on these practices. We published an overview paper, "Innovation as Language Action," ACM *Communications* (May 2006), 47-52.

[3] P. J. Denning. "Mastering the Mess." ACM *Communications* (Apr 2007).

their commitment. I believe that Roberts' 7 factors are all symptoms of the obsolescence.

This is the same idea as was examined by Adrian Slywotsky in his book *Value Migration*.[4] There he discussed the paradox of innovative businesses going out of business because their customers found other business designs more attractive. In an education context, "operating model" seems like a better term than "business design". I believe that the migration of students away from computer science signals that our "operating model" for computing education is obsolete. We need a new operating model.

Through this project, I propose to explore the design of a new operating model for computing education. The main project objective is to show what such a framework might look like, and then by working with the ACM Education Board, promulgate it and get people to try it out.[5] In the process, we will identify and support new leaders within computing. Even though the situation is complex, by applying pressure to the right leverage points we can move the system. In the next section I'll discuss three leverage points and a strategy to mobilize around them.

Andrew McGettrick, currently co-chair of the ACM Education Board, has provided his endorsement. The ACM will be key to providing support structures to sustain new operating models and nurturing a new generation of leaders to carry them forward.

Sketch of an Alternative Operating Model

The current operating model for CS curricula is based on four tacit assumptions. (1) Departments teach standardized courses as recommended by the ACM and IEEE, which update their recommendations about once a decade. (2) The recommendations derive from a body of knowledge arrived at by a community consensus process. (3) The body of knowledge is expressed as a list of core technologies that all computer scientists should know. (4) Curriculum effectiveness is evaluated by academic achievements of students and quality of faculty presentation in the classroom.

These four assumptions are linked to the 7 factors cited by Roberts. Standardization is a problem because the curriculum is much the same everywhere, so when prospective students come to believe the standard curriculum is unattractive, there are no alternatives, and thus everyone experiences a decline. Consensus is a problem because it takes too long in a large and dynamic field, many resulting recommendations are out of date by the time they are made, and few members of the field are completely satisfied with the results. Basing the body of knowledge on technologies is a problem because

---

[4] Slywotzky, Adrian. *Value Migration*. Harvard Business School Press (1995). I commented on the application of these ideas to the university in "Business Designs for the New University," *Educom Review 31*, 6 (Nov-Dec 1996), 20-30.

[5] I have close working relationships with the Education Board. I chaired the board from 1998-2004 and have been a member since.

computing technologies are constantly changing and thus we need a new consensus every 5-10 years. Assessing students by GPA is a problem because in the real world they will be judged by the value and results they produce for their customers and employers; success with customers depends on a skill of listening for and delivering value, which is not taught by classroom lectures and tests. Assessing faculty by peer review of classroom practice and by student opinion polls is a problem because it does not encourage faculty to be good coaches.

I propose to flesh out a model based on four alternate assumptions: (1) diversity, (2) individual department initiative, (3) body of knowledge expressed as fundamental principles, (4) student achievement measured by their success in projects and faculty teaching achievement by their success as mentors and coaches. This table compares with the traditional operating model:

Traditional CS Model	Possible New CS Model
Standardized curriculum	Diversity of curricula
Consensus process to agree on curriculum	Individual departmental initiatives and innovations
Body of knowledge a taxonomy of core technologies	Body of knowledge a taxonomy of great principles
Students assessed by course achievement, faculty by student opinion surveys	Students assessed by project achievement, faculty by project and life coaching

For the CS&E departments who participate, this new model is likely to stimulate innovation, make them more attractive to study in, and give students a range of choices about departmental emphases and styles.

This approach is consistent with the two key principles noted in the CSTA commentary. (1) It will allow students a variety of ways to play in big games involving computing by engaging with exciting applications. (2) It will allow students to look to their computing teachers for adult leadership and coaching.

To arrive at this goal I will organize three invited workshops and then write a report about the model and recommendations for encouraging departments to participate. For each workshop, I will organize a small steering committee of junior and senior experts to help plan the schedule and determine the invitation list; I will use my local (Cebrowski Institute) administrative staff for conference support. Because one of the overall purposes of the workshops is to nurture new leaders, we will ensure a strong representation of junior members of the field in the steering committees and in the workshop attendees.

After the following summaries of the workshops and the final report, three sections give more details on each of the workshops.

Workshop 1: Great Principles of Computing

In the first year, the workshop on Great Principles of Computing will bring completion to a project to develop an articulation of our field in terms of fundamental principles. The field is now mature enough to do this. We now have a preliminary division of the principles into seven categories, detailed drafts of principle statements in each category, and a few commitments from leading experts to write principle stories to start a Great Principles Library within the ACM Digital Library. The result of this workshop will be a statement of the great principles of computing, an action plan for recruiting leading experts to write principles-stories for a Great Principles Library, a plan to instantiate the GP Library within the ACM Digital Library, and an editorial process to maintain and update the GP library.

The Great Principles formulation offers an alternative to the current technology-based expression of our field. It helps people locate or discover principles in the complex arrangements of technologies. A great principles framework can evolve as new principles are discovered and older principles diminish in importance. This is why the GP Library has an editorial process to review and update its contents. ACM can maintain the GP Library as part of its records of the computing body of knowledge.

Workshop 2: Stimulating Innovation in CS&E Departments

In the second year, the Stimulating Innovation workshop will recognize innovators and examine ways to support them in their innovation and teaching practices. The participants will be CSTA and college faculty who are already demonstrating innovations. We will also bring in others who have innovative approaches to turning kids on about science (such as Shawn Carlson, founder of LabRats, labrats.org, and a representative of USFIRST, who use Lego MINDSTORMS to organize student robot competitions). The workshop objectives are to recognize innovators and publicize their innovations, to initiate within ACM an innovator's portal of innovation descriptions and guidelines for other departments to imitate, to develop a support network for curriculum innovators, and to help faculty learn to be effective coaches for their students. By recognizing innovators, the workshop and follow-on sustaining processes will nurture new leaders.

After the workshop, the steering committee will develop a recommendation to the ACM Education Board on how to sustain a process of ongoing curriculum innovation and recognition for innovators; and a recommendation on how interested faculty can become excellent coaches. (Details later in the Discussion section.)

Workshop 3: Project Based Learning

Also in the second year, the Project Based Learning workshop will give its participants an immersive look at a living, working, successful alternative operating model. The model has been developed and implemented by Neumont

University in Salt Lake City. The workshop participants will see the curriculum in action, meet the faculty and students, and discuss how the structure can be scaled and transferred to other departments who are interested in project based learning. They will discuss how to organize a project-based curriculum to avoid leaving students with knowledge gaps. Neumont University has offered to host this workshop. Neumont's president, Graham Doxey, provides an endorsement.

Final Report

The final report of the project will be a description of the principles framework, an innovation support plan, guidelines and support for project based curricula, and recommendations to the ACM Education Board for sustaining these three activities. We will work with the Education Board all along to review drafts and start bringing ideas into play as soon as they are ready.

We do not expect the Education Board to consider the results of this project to be the only possible new operating model; we expect them to be interested in encouraging experimentation with other models as well.

Lasting Impact

The ACM Education Board has already concluded that the current curriculum process is too slow and burdensome. That process is built around a community consensus on a body of knowledge (BOK) and core curriculum recommendations. Given the number of players in the field and its complexity, it takes 7+ years to complete the cycle (CC2001 took from 1999 until 2006). I have supported a simpler process wherein the Board manages the computing BOK and disseminates grass roots generated guidelines on specific aspects of computing. Individual departments would not be pressured to teach the whole BOK in their curricula, only to be true to the principles and to inspire their students to be interested in them. The BOK would be maintained in multiple forms including the current technology list, an ontology format (under development with NSF support), and the great principles form.

To obtain a lasting impact of a new business model, we need to put into place appropriate support processes.

- For the Great Principles Library (part of the ACM Digital Library), an editorial process that maintains the library (a set of principles stories and source materials) and incorporates worthy new additions.
- For stimulating innovation, three support processes. One is an Innovator's Portal (part of ACM Digital Library) that profiles education and technology innovators and their innovations, offers guidelines for departments that may wish to their practices into curriculum, and provides access to affinity groups in innovation. Second is a Guidelines Library that offers recommendations for implementing specific ideas in a curriculum. An example might be a computer security guideline for adding a security track to a curriculum; only departments interested in teaching computer security would be interested in the guideline. Third is

an ongoing Innovation Practice workshop, possibility held in conjunction with the annual SIGCSE symposium, to offer faculty practice in the foundational skills of innovation and coaching.

- For project based learning, at least one support process. I will help Neumont University and ACM Education Board cooperate on workshops that will help interested faculty from other departments get started in adopting project-based learning. We will need to consider additional support functions, such as a Projects Library along the lines of the "nifty assignments" that have become so popular in SIGCSE.

In addition, I will continue my writing and speaking on particular topics (IT Profession columns, SIGCSE, CACM, and other venues) and will complete books on innovation and on the great principles of computing.

Discussion of Great Principles

What is computation? How does it work? What are its limits? It is a science?

Our traditional approach to answering these answers is to explain the core technologies of computing and then describe how they fit together to make up computing systems. It was a good approach during the early days when the core technologies were few: algorithms, computation models, compilers, and logic circuits. Over the years, many core technologies have been added, including chips, operating systems, databases, networks, intelligent systems, robotics, graphics, and more. Today the total number of core technologies is around three dozen. For the newcomer, learning the inner workings of 36 technologies and their 630 possible direct interactions presents a daunting challenge.

An alternative approach is to focus on the principles that define and constrain all computation. A principles-based approach is followed in the mature sciences such as astronomy, life sciences, chemistry, and physics. These sciences offer rich structures built up from a few base principles. The computing field has only recently reached a level of maturity where it can do the same.

The Great Principles project aims to develop a principles articulation of the computing body of knowledge. The benefits of doing this are:

- Expose the deep structure of the field. Doing so can reduce the apparent complexity of the field, contributing to greater understanding, better designs, and simpler, more reliable systems.
- Enable designers and users to see the common principles on which computing technologies are based. This will facilitate sound designs and cross fertilization among technologies.
- Establish a new relationship with people from other fields by overcoming the perception that we have few principles of our own and instead borrow heavily from other fields; and by reducing the tendency to hype computing technologies toward expectations they cannot meet.
- Provide inspiring stories about the development of the field and its principles for young people.

Despite six decades of growth and accomplishment, our field is still widely regarded by professionals in other fields and by the public as immature. We struggle with the perception that we are mainly a technology field that applies principles borrowed from mathematics, physics, biology, cognitive science, electrical engineering, and systems engineering. In reality, the people of computing have discovered an extensive array of original, fundamental principles beyond Moore's Law. These perceptions hurt our relations with other fields and recruitment of new scientists and engineers to our field.[6]

Opening to the Future

A great principles approach is a different way of thinking for our field. It complements and extends our traditional ways. It enables our understanding to penetrate the complex mazes of technology that so often confront us, seeing through to the principles that guide and constrain the technology. It also enables us to see connections with other technologies based on similar principles. It thus opens the door for new discoveries.

The principles we work with today were discovered as past generations of computing people grappled with complex, seemingly intractable problems.[7] Their discoveries led to solutions of these problems. But the process of discovery is hardly over. We grapple today with a new array of seemingly intractable problems, and we will surely discover new principles that will enable their solutions.[8] Over time, many of the older principles may fade from importance while the newer ones move into the spotlight.

The preliminary listing of principles illustrates these dynamics. Some of the principles listed there are relative recent discoveries. For example: lossy compression based on preserving valued bits underlies MP3 and MPEG; networks that send only bits valuable to the receiver avoid information overload; heuristics offer good-enough solutions to many intractable problems; hyperlinks create a rich semantic Web in the Internet; intelligent systems mimic human behavior without requiring us to know whether the brain is computational; frequent customer interaction improves the design of complex software.

Thus a great principles approach to the field does not foreclose the discovery of new principles; it encourages discovery.

---

[6] P. J. Denning. "Is Computer Science Science?" ACM *Communications* (April 2005), 27-31.

[7] Examples: noncomputability, code breaking, ballistic orbital calculation, thrashing, timing bugs in parallel systems, secret communication in open Internet, fast-enough algorithms for common problems, sharing information, data compression.

[8] Examples: user interfaces, identity theft, securing networks against attack, spam, information overload, dependable software, hastily formed networks, distance learning, discovering terrorist plots beforehand.

The GP Project

I started exploring the idea of a great principles formulation of the field in 1998 at George Mason University, where I designed and experimented with a capstone course around this theme. At the Naval Postgraduate School, which I joined in 2002, the CS faculty adopted the Great Principles theme in 2003 and built a major revision of the CS curriculum around it.[9] Our great principles course is used by mainly graduate students entering our program without a bachelor's degree in computer science: it offers them a rapid orientation and road map to the field. In a keynote speech at SIGCSE 2004, I reported on our experience and proposed that others consider this approach.

With help from Jim Gray and others, I put together a task force of the ACM Education Board to lead to the development of a Great Principles materials library as part of the ACM Digital Library.[10] The GP Library will be a living representation of the principles of the field. Its editorial oversight process will respond to the dynamics of the field, incorporating new discoveries and subsuming older principles into the newer.

Project Web Site

The project web site, http://cs.gmu.edu/cne/pjd/GP, contains:

- A justification for dividing computing principles into seven categories: computation, communication, coordination, recollection, automation, evaluation, and design.
- Drafts of principle statements within each category.
- A prototype of a principle story (Locality Principle) to serve as an example for other stories now being written.
- An FAQ about great principles definitions and approach.

Project Status

In 2005, the ACM task force agreed on a division of principles into seven categories. They agreed on a medium-term objective of a library of principle stories -- narratives and other materials about the discovery, development, and applications of fundamental principles. The library and its editorial oversight would support the objective of keeping up to date with new discoveries. Several task force members agreed to draft stories about principles that were central to

---

[9] P. J. Denning. "Great Principles of Computing." ACM *Communications* (November 2003), 15-20. A description of how we brought the framework into the NPS CS curriculum was "Great Principles of Computing Curricula," *Proc. SIGCSE* 2004, 336-341.

[10] The task force members, who serve as a steering committee for the project, are: Bob Aiken, Gordon Bell, Fran Berman, Fred Brooks, Jeff Buzen, Fernando Corbato, Peter Denning (chair), Ed Feigenbaum, John Gorgone, Jim Gray (co-chair), David Gries, David Harel, Juris Hartmanis, Anita Jones, Mitch Kapor, Alan Kay, Peter Neumann, Paul Rosenbloom, Mike Stonebraker, Andy Tanenbaum, Allen Tucker, and Moshe Vardi.

their own careers.[11] Others were hesitant to spend time writing up a principle in the absence of a larger community agreement that the principle is fundamental. They suggested that a workshop would be needed to discuss the principles and then cast a wider net to get commitments on stories. That suggestion inspired the present proposal for a workshop.

In summary, when this project is done we will have:

- Taxonomy of Great Principles;
- GP Library consisting of principle stories and other source materials;
- Process to maintain, evolve, and improve library; and
- Advice for departments who want to use GP as curriculum theme.

This is a low budget project with high payoff when done. The proposed workshop will considerably accelerate the completion process.

Discussion of Innovation Stimulation

The purpose of this workshop is to devise ways to stimulate curriculum innovation in computing departments. Curriculum innovation means not only new ways to teach computing, especially by involvement in exciting applications, but new curriculum structures, new organizing themes, etc. The participants will be curriculum innovators.

The workshop will initiate work toward three objectives to support curriculum innovators:

- *Innovator Recognition.* Curriculum innovation starts with individuals who see new possibilities and are willing to prototype them and then work for their adoption. We will call attention to these people and encourage others to imitate them. Being invited will be a mark of distinction. The workshop steering committee will reach out for nominations through personal networks, the ACM community web site, the SIGCSE community, and the CRA Forsythe list of the PhD granting departments. The ACM PR office will help recognize and profile the workshop invitees as the top curriculum innovators in the field.
- *Innovator Portal.* This part of the ACM online community will support the Innovator Recognition theme with a library of profiles and guidelines that would help others adopt their innovations. It will also support discussions around innovative ideas.
- *Innovation Practice.* This part of ACM Professional Development will support the teaching practice of innovative educators with workshops on the foundational practices of innovation and coaching.

---

[11] Len Kleinrock (resource sharing), Jeff Buzen (transaction processing model), Jim Gray (storing and retrieving data), Ed Feigenbaum (representing knowledge to automated cognitive tasks), Allen Tucker (programming languages), Eric Roberts (errors in software and programming), and myself (locality).

The workshop will support these elements as follows:

1. Give each participant some time to present their innovation to the group.
2. Discuss the obstacles they face and how they might overcome them.
3. Discuss the foundational skills of innovation and how the participants can use them to advance their projects back home. We would include exercises in the areas where the obstacles are greatest. (The foundational practices are drawn from our innovation project.[12])
4. Discuss what is involved to be a great (life) coach. This segment would be facilitated by someone with an established reputation as a great coach. (I know several top people whom I can invite.)
5. Discussion of the Innovator Portal and Innovation Practice workshops.

The workshop outcomes will include:

1. ACM PR to recognize and profile the workshop invitees as the top curriculum innovators in the field.
2. A plan for the ACM Ed Board to create the Innovator Portal.
3. A recommendation to ACM (and SIGCSE) about Innovation Practice workshops to stimulate and support innovators. This may also include a recommendation to NSF about support of these workshops.

Discussion of Project Based Learning

Project based learning means a learning environment organized around students working together on projects, learning what they need as they go along. This environment immerses student teams in software developments with customers. Computer science is a rich field for such an environment because of the enormous possibilities for applications in many other fields. Computer science educators have discussed this mode of learning at length, but there have been very few successes because of the work involved to move from lectures to projects, avoid gaps in coverage of fundamentals, and maintain a mapping from accreditation requirements to courses.

Neumont University was founded in 2002 in Salt Lake City to provide an undergraduate BSCS degree strongly emphasizing software development. It is a private university whose income is solely from tuitions. They designed a project-based curriculum from the ground up. The curriculum moves students through three stages of projects: foundational, faculty-directed, and external customer driven. They also organized their schedule so that the entire undergraduate program is covered in 8 intensive quarters -- a Bachelor's degree in two full years. They now have 300 students enrolled and a growing applicant pool. In 2006, their graduates received an average of 2.5 offers with median salary $60K. Their approach is immensely attractive to employers, from large Fortune 500

---

[12] P. J. Denning and R. Dunham. "Innovation as Language Action." ACM *Communications* (May 2006), 47-52.

companies to small businesses, who provide meaningful project work and hire Neumont graduates.

Three principles guided the design of their program. (1) They worked backward from customer (potential employer) requirements to the curriculum. From education research they knew that 75% of what students retain comes from doing it in projects. In their curriculum, students spend 80% of their time in projects, doing and learning together. (2) To avoid gaps in knowledge they staged their projects to move from foundations (repeating experiments of others) to projects with faculty as customers and then to projects with outside customers. (3) The faculty play strong roles as adult leaders, mentors, and coaches to assure all students are learning and have no knowledge gaps. With this design, their students are definitely in contact with exciting applications and under the mentorship of coach-teachers.

I was a consultant during their curriculum design phase. I have been very impressed with the design and ingenuity of its operation. The design was hard, intensive work and it required many adjustments to get it right. But their work paid off. They have numerous applicants and have received recognitions and awards. Now that they have a working model, we have the opportunity to revisit the project based curriculum idea and figure out how to transfer it into traditional institutions that have been interested in this approach for a while.

A Concluding Note for Proposal Evaluators

I understand that some reviewers may have a bias toward recommending emerging junior leaders as CDEF recipients, rather than established senior leaders such as myself. I do not submit this proposal to promote my own leadership development, but to promote the leadership development of the two dozen or more junior leaders who are likely to participate in the workshops. This proposal involves both junior and senior leaders in all the workshops and provides for the design of support structures that will especially benefit junior leaders. If you recommend for this project, you will get a PI committed to nurturing a new generation of leaders that can make the computing field an attractive home for new generations of students.

All this is consistent with the NSF guideline for CDEF projects, which states: "CDEF grants will recognize accomplished, creative, and talented computing professionals who have the potential to serve as national leaders or spokespersons for change in undergraduate computing education. CDEF awards will be made to individuals who have achieved distinction in the computing profession, who are committed to transforming undergraduate computing education, and who have innovative ideas on how to do so. CDEF recipients may spend significant time and effort on projects focused on innovative, original, and possibly untested ideas that will benefit undergraduate computing education on a national scale."