The Road: Reinventing Education

David West
College of Santa Fe
Las Vegas, New Mexico USA
profwest@fastmail.fm

Pamela M. Rostal
Perficient, Inc.
Minneapolis, Minnesota USA
pam.rostal@perficient.com

Eugene Wallingford
University of Northern Iowa
Cedar Falls, Iowa USA
wallingf@cs.uni.edu

Joseph Bergin
Pace University
New York, New York USA
jbergin@pace.edu

Robert C. Duvall
Duke University
Durham, North Carolina USA
rcd@cs.duke.edu

Rick Mercer
University of Arizona
Tucson, Arizona USA
mercer@cs.arizona.edu

Richard P. Gabriel
IBM Research
Hawthorne, New York USA
rpg@dreamsongs.com, rpg@us.ibm.com

Abstract

We propose a vision—an idea for an educational setting and program that dramatically expands the breadth and depth of knowledge—including a truly meaningful integration of the liberal arts—expected of a computer science / software engineering graduate. Roles and relationships are redefined to promote the emergence of a community of practice, one that will attract women and other under-represented groups in a uniquely powerful way. We offer some concrete means for realizing our vision, derived in part from a pilot program established by two of the authors and a recently initiated program. Instead of conclusions, we offer expectations of results from successfully implementing our vision.

Imagine...

...a studio in Renaissance Florence; a master and advanced apprentices at least; several arts being worked shoulder to shoulder: sculpture, painting, goldsmithing, even poetry with masters for each; a spectrum of younger apprentices eager to master one of them but eager also to learn another, or two. This the ideal of the *bottega*:

- a "storefront" where goods and services are produced and delivered to paying customers
- a workshop simultaneously engaged in the craft, in building the tools and discovering the techniques that advance and support the craft, and teaching that craft to apprentices
- a place noisy with multiple projects and activities; walls and benches covered with works in progress and exemplars of the craft
- a place filled with the tools of the craft (add computers and digital displays to the easels, brushes, hammers, chisels, carving, forges, kilns, model making, etc. tools found in a typical bottega); with room for lounging and eating facilities as well
- an intellectual center that is a "must visit" for masters, scientists, and thinkers visiting the area, overseen (deliberately avoiding the term managed) by local masters and journeymen
- an environment and atmosphere that is very self-consciously multi- and inter-disciplinary; that mixes theory and practice almost without differentiation
- a place full of music, especially "after hours"
- a place to share food and drink (and perhaps sleep)
- a fountain of innovation and creativity

Now imagine a computer science education based on this model. All learning would be based on bottega projects—software projects executed in teams—each project bristling with a set of required skills and knowledge that when learned (at various depths and degrees) educate the student. In the bottega there would be few if any "lectures"—instead there would be brief expositions, explorations, reading, readings, and web searches—with a preference for hands-on learning or learning in a context, typically for a purpose, for an particular audience.

Thinking about such an approach to education, the questions that arise include these:

- how well can undergraduates learn / master computer science and/or software engineering "taught" this way?
- can the education of a future practitioner take place in the same room as the education of a future researcher?
- can an entire computer science undergraduate education—including majors and electives—be taught this way?

Our Proposal

We believe it is possible to produce a completely educated computer scientist by using the bottega model. We believe we can achieve this using the following ingredients:

- around 10–30 students of varying experience inhabit a bottega, including advanced undergrads and grad students
- around 2–6 mentors at any given time rather than "ordinary" teachers guide the activities of the bottega; the mentors are masters of their crafts and disciplines, and include, at various times, mentors from all the departments in the university
- each bottega is an open studio with dormitory rooms, a kitchen, and common areas, so that the students are always nearby
- students spend 4 years in the bottega, 8 hours a day, 5 days a week; all instruction/learning aside from field trips takes place in the bottega

OOPSLA'08, October 19–23, 2008, Nashville, Tennessee, USA.

- students execute projects as the medium for learning; each project has a set of available "opportunities for learning" which, taken over a four-year period, cover not only all the computer science and software engineering topics, but also all subjects to the depth required for minors in other disciplines and breadth of education (we will provide an example later in the paper)
- each bottega is a community of learning, with the roles of teacher and student or mentor and apprentice fluidly morphing into each other depending on who is more knowledgeable or accomplished at what
- visitors—including researchers, mentors, and shorter term students—invigorate the bottega with new and different points of view; not every visitor is a computer scientist, with people from different disciplines and practices encouraged

Each individual bottega represents a continuous 4-year education using multiple projects. The opportunities for learning represent fractional units in a traditional unit-based, semester-oriented arrangement of courses and classes. These atomized bits of knowledge and skill are assessed on a 7-level scale:

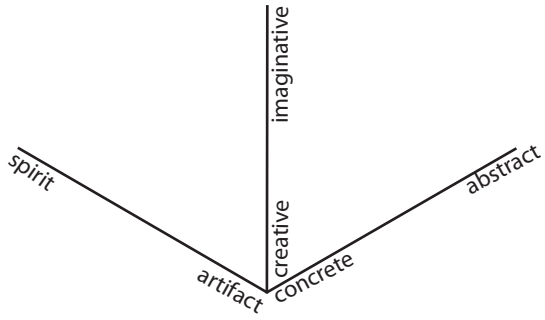
1. has mastered the vocabulary
2. can do it with help
3. can do it alone
4. can do it in a novel context
5. can mentor or teach it
6. can create teaching materials for it
7. can make a novel contribution in the area

As a student progresses through the bottega, he or she is assessed on this scale to judge whether sufficient progress is being made toward the degree.

The essence of how a computer science education works under this model is this: learning happens when you *do* under supervision while *reflecting* on what you're doing, and both within CS and without, by enough just-in-time learning experiences. This is how any apprentice-based education takes place, and also how most fine-arts education is structured.

Some Background Stuff

Many of our undergraduate computer science programs were triggered by if not aimed at educating practitioners. The impulse to educate practitioners has had the effect of creating an educational approach that attempts to educate both the practitioner and future researcher at the same time using roughly the same courses—though there are courses that distinguish between the two groups. To properly educate a researcher, topics in programming languages, for example, should take a critical and perhaps even historical approach, so that it is clear that the current state of the art is simply a point along a (possibly evolutionary, if we're lucky) path, while for the practitioner, teaching programming languages should be aimed at achieving a thorough competence at designing for and programming in those languages, which requires, we believe, also being knowledgeable about the principles behind the mechanisms, if not expert. Because we don't aim at either extreme of student—neither future researcher nor future practitioner—we are serving neither particularly well.



An aspiring researcher should emerge from an undergraduate education with the sense that there is much unfinished business, and that perhaps even some of the seemingly finished business is actually provisional. An aspiring practitioner should feel at home with and have great confidence in the taught tools, and should be equipped to learn new ones with a certain ease. The mathematics major—at least when some of us were undergraduates—was broken into applied and pure math. Someone in applied math was interested in using mathematics at an expert level for some purpose other than mathematics itself. Many of the courses such a person took would be the same as the pure math major. Someone in pure math was interested in how mathematics proceeds with an aim toward working on hard problems after an advanced degree. During his or her education, an applied math major practices using mathematics while a pure math major practices proving theorems.

By taking the particular form of middle course we in computer science education have been taking, we achieve mediocre results for all.

What Kinds of Graduates are Produced

This approach to education—the bottega approach—does not produce ordinary workers such as plain-old mechanical coders. It is designed to invigorate an information-based economy. To inform our design of the approach, we spent some time looking at the range of people who would come out of such a degree program. The diagram at the top of the page defines the range of characteristics we identified.

The first (y-axis) is the ***creative–imaginative*** axis. It represents how within-the-box a person's creativity lies. Someone near the origin would be able to work with the materials at hand by combining elements in novel ways when appropriate. To use a poetry example, a creative poet is able to operate in the usual creative ways expected of a trained, talented, and accomplished poet. He or she might well win prizes for their work, but certainly either is or would at some point be published. An imaginative poet would be able to invent new genres of poetry or work in unexpected ways. Moreover, his or her work would be considered not only surprising but groundbreaking. An example of a pair of such poets would be the guy who invented *flarf* and those who come to practice it. As defined by Wikipedia [ref], practitioners of flarf

> *practice an aesthetic dedicated to the exploration of "the inappropriate" in all of its guises. Their method [is]*

to mine the Internet with odd search terms then distill the results into often hilarious and sometimes disturbing poems, plays, and other texts.

The poet who came up with flarf would be considered imaginative. A poet who, on learning of flarf, quickly becomes fluent in it, producing nicely hilarious and sometimes disturbing flarf poems would be a creative poet.

The second axis (x-axis) is the ***concrete–abstract*** axis. It represents the sort of material the person works on and with. Someone working with or creating source code is near the origin, and someone who is working with or creating abstractions or ideas is at the abstract end. To use a software example, a creative builder—who is at the origin and on the x-y plane—would be someone who produces very elegant, maintainable, and cleverly put together code, often in surprising and novel ways. An *ordinologist* would be at the origin on the creative–imaginative axis and at the abstract end of the the concrete–abstract scale. An ordinologist is someone who takes a discipline, characterizes its subfields, understands the relationships between them, and produces a catalog or encyclopedia of the results. Ernest Bielstein who wrote the 60-volume "Handbook of Organic Chemistry" is an example. Peter Wegner, when he wrote his categorization of the various variants of object-oriented programming, "Dimensions of Object-based Language Design," [ref] was acting as an ordinologist.

The third axis (z-dimension) is the ***artifact–spirit*** axis. It represents the sorts of effects a person is aiming at. At the origin, a person is trying to produce an artifact that will make the work or life of its users enjoyable and comfortable. About ⅓ of the way out, the person is trying to write a paper that will make the life of practitioners and theoreticians easier. About ⅔ of the way out, the person is trying to write an Onward! paper, which is intended to change how people think about computing or something similarly grand. And all the way out, a person is trying to change the nature of reality or of the spirit through whatever means are at hand.

It is illustrative to look at the career of Christopher Alexander, the architect as a way of understanding this space. When he was acting as an actual architect, he was operating close to the origin: creative, working on actual buildings, and just trying to create artifacts (the buildings). In *Notes on the Synthesis of Form*, he was operating in the abstract, producing a book that would influence his and related professions (and hence somewhere back on the spiritual axis), and from the first part of the book to the last, he moved from creative to imaginative. With *The Timeless Way of Building*, he became fairly imaginative, fairly spiritual (he was trying to change how architects think about the way they work), and abstract. With *A Pattern Language*, he became mostly purely creative, abstract, and only mildly spiritual (as if he had written a paper). And finally with *The Nature of Order*, he maxed out on being imaginative, spiritual, and abstract.

As an architect—the first stop on his journey—Alexander was a very concrete kind of guy. He was interested in buildings that would bring joy and comfort to its inhabitants. Other architects—such as Frank Gehry designing the Stata Building (CSAIL)—are trying both to create an ordinary building as well as to express an abstract aesthetic point of view. Gehry is a middling imaginative. This puts him about at the center of the imaginative–abstract plane, and a tiny bit back from the origin on the spirit axis—because he is trying at least a little to influence how people view the built world.

A computer science researcher could be anywhere on the imaginative axis, but probably toward the imaginative end, somewhere pretty close to the abstract end of the abstract axis, and somewhere at least ⅓ of the way back on the spirit axis.

This categorization, as does every dimensional categorization, presents the opportunity for unintended value judgments. In most 2-dimension displays, for example, the upper righthand corner is considered the best, most highly valued place to be while the origin represents primitive or rudimentary aspects of the dimensions. If you have 0 of something, it must be bad. This is reinforced by businesses that often present their profit or revenue projections in the form of hockey-stick-shaped graphs with money represented on the y-axis while time marches along on the x. In our minds, no point in this octant is a better or worse place for a person to be. Being creative is just as wonderful as being imaginative; working in the concrete is just as valuable as working in the abstract, and producing artifacts is as important as producing visions.

This categorization gives us a way to look at the kind the people we want the bottega approach to produce. The axis that is clearly incomplete is the imaginative one, because there certainly are people who are not creative at all, but our octant ignores them, as perhaps does our approach.

Example

In the bottega, the curriculum consists of several classes of problem that collectively cover the topics students need to learn. Each class of problem requires a specific set of skills to solve. By working on a problem instance from a given class, students gain experience with the corresponding skills. During their time in the program, students will work on any number of problems that are instances of each class, honing each skill until they have demonstrated competency or mastery. Each skill consists of several "knowledge atoms", which correspond to something like the knowledge units of the ACM/IEEE 1991 joint curriculum recommendation. [ref] In designing this curriculum, we would consider various model curricula from the ACM and IEEE, to ensure that the set of problem classes covers the knowledge units the professional societies—and we—deem necessary.

One class of problems that might occur in the curriculum is the planner. Many systems take as input a set of goals and a set of constraints, and produce as output an arrangement of actions that meets the goals while satisfying constraints. Instances include scheduling an academic department's course schedule for a semester, developing a schedule of experiments for a scarce resource such as the Hubble space telescope, and helping a student to create a plan of study. A *student advisor* system arranges in time a set of specific courses or learning moments that matches the individual needs of the student while constrained by the availability of courses. The system must provide the ability to modify the plan as circumstances change, such as a change in the availability of instructional resources. It should also support "what if…?" scenarios, enabling the student to explore the implications of particular desires manifested as constraints. Finally,

it should support various look-up functions, in particular the ability to view a degree audit that lists which requirements have been completed and which remain to be satisfied.

Implementing a student advisor, or solving any problem in its class, will require the developers to address a number of smaller-grained problems: planning, accessing a database, propagating constraints, optimizing results, receiving and processing input, and presenting results. Solving these lower-level problems confronts the student with a need to work on a number of specific *opportunities for learning*:

- **software engineering practices**: This problem places special emphasis on expressing requirements, in the form of both task requirements and the domain content requirements dealing with courses and programs of study. Students will need to learn methods for gathering and analyzing requirements, whether agile or more traditional, and also methods for encoding the requirements of all degrees and programs offered.
- **tool selection**: Students will have to choose one or more programming languages, an IDE, and perhaps a set of libraries to use.
- **artificial intelligence**: The system's goals and constraints must be encoded in a way that enables them to be reasoned over in different ways. Students can explore several alternative approaches for solving the problem, such as goal-based search, rule-based reasoning, constraint satisfaction, and perhaps even metaheuristic techniques such as simulated annealing. The system will need to support optimizations in the dimensions of time and money, which offers opportunities to explore yet another class of algorithms.
- **user experience**: Students can develop a user model and from it craft a user interface that supports it.
- **databases**: The student developers will need to build a data model that embodies information about courses and program requirements. From it, they can implement a database that enables the system to extract information about courses based on the student's program of study and other constraints.

Not every student will work on the same opportunities for learning to the same depth when doing a project of this sort. Students will work on this project at different points in their development and with different interests. As a result, they will focus on some opportunities for learning, work on others at a shallower level, and perhaps pay little attention to another set. By doing a number of different projects over time—from this class of problems and others—students will have opportunities to encounter each opportunity for leaning again in a different context and work on all of them at a level necessary for demonstrating proficiency.

Projects do more than confront students with the need for skills and knowledge specific to the discipline. More generally, building a system such as a student advisor also gives students experience with *group skills* that transcend any particular project. These include teamwork, coaching, leadership, and community values. Every project incorporates these skills. Novice students will be just beginning to learn how to work on a team and to live the values of the community. As they mature, they will take on more leadership responsibilities and contribute more fully to their teams. By constructing teams with a mix of novice and mature students, each type of student has the chance to practice the skills it most needs and to learn from one another.

Every class of problems also provides hooks to many *generic learning goals*, such as formal reasoning, abstraction, and written communication skills. Sometimes, a particular opportunity for learning required for the project contributes in an immediate way to a generic learning goal. For example, the developers of a planning system have an opportunity to become more autonomous as they learn to consider alternatives and challenge conventional wisdom. While much software is constructed with a top-down control structure, a constraint satisfaction system can be constructed as a collection of independent constraint objects, interacting to solve the problem with no master module. Similarly, developers of a student advisor system in particular will need to develop an awareness of all the disciplines that are represented in the course catalog. The act of gathering requirements and implementing ways for the system to make choices among courses provides the student with the opportunity to learn more about each of the disciplines. In a sense, the full set of courses and their descriptions in a typical university catalog is a map of human knowledge, and knowing that map is a step toward learning that material. And in this way, the project contributes to the student's liberal education.

In other cases, the project contributes to generic learning goals through more generic means, such as requiring students to seek alternative resources and learn in a self-directed manner. Finding alternatives can be in the form of consulting mentors or other students, or in the form of finding code that helps solve the problem. In this case, the students can be encouraged to study the software they have found, explore the underlying domain or discipline knowledge, critically evaluate the code and the use of the programming language by its author, and otherwise engage in critical thinking about what's been found. Short essays on these topics can help students gain strong mastery of the craft elements in their major (computer science) as well as learn design and implementation techniques from "the masters."

Writing these short essays well requires understanding how to write, composition, rhetoric, and a host of other non-computer-science skills. Mentors from the English department or the creative writing department can provide project-based learning opportunities for the students in the bottega.

Early in students' careers, mentors will provide more guidance; over time, they will shift more responsibility for learning to the students. Arching over all other goals, mentors will guide the students as they integrate the many lessons they learn on the project into their personal and professional toolboxes.

Generic learning goals persist from project to project, regardless of problem class. Over time, students grow in skill and facility as learners, communicators, team members, and leaders. No matter their current level of achievement, they are able to learn more, as well as to help more junior students grow.

Finally, the act of working on a particular development project opens the door to opportunities to learn some topics in more depth and to learn about areas within and beyond computing. The student advisor project might lead a student to study the area of natural language translation, in order to understand how the system might better infer from the on-line catalog whether a particular course meets a particular curricular need. This might lead to a wider discussion of linguistics and how processing natural language differs from processing programming languages.

While implementing the system, the student may become curious to learn more about object-oriented language theory or systems control theory. The student may look even wider, growing interested in one or more of the subject areas described in the course catalog. This could lead the student to read about Shakespeare's effect on modern literature or the relationship between cognitive and social psychology.

Such side tours can be either to explore recent research in an area—such as looking at how to encompass a number and variety of databases under the umbrella of a unified virtual database in the example at hand—or to do some research to move the frontiers out or understand a subject more thoroughly. Students aiming at a spot in the octant that represents a researcher will take such side tours regularly, and increasingly as they mature in the bottega.

The domains for some projects provide the opportunity to learn about those domains. For example, for a project that is simulating parts of the biological world, there is the opportunity to learn as much biology as makes sense to understand the requirements for the simulation. Students pursuing minors in biology would then take the time to pursue those studies. Using assessment, the mentors can determine how thoroughly a subject—such as a minor subject—has been mastered.

With proper guidance from mentors and subject-area experts, working on a student advisor system can contribute to many parts of a liberal education, and with a carefully chosen series of projects, the entire undergraduate education can be provided.

Community Learning

Although many organizations focus on skills development, very few consider community learning an integral part a person's skill set. People in a team setting, whether in academia or industry, engage daily in serious discussion yet often encounter problems arising because our social, cultural, and educational backgrounds have not prepared us to be good at discussing. Our bottega approach depends on the mentors and students developing excellent discussion skills, discussion being the primary avenue for both getting the projects done and learning in general.

The bottega will use techniques such as those developments by the Touchstones Discussion Project [ref] to enable participants to correct problems that plague learning groups—accepting contributions by non-experts, cultivation of collaboration and active listening skills, and emergence of participative leadership.

The technique is based on discussing readings in a formal or semi-formal setting. In this it shares some common elements with the writers' workshop, which is used in fine-arts education. Although the content of the readings can be relevant, the emphasis of meetings is on the process of discussion. By varying communication opportunities—from an individual filling out a form to small groups of 2–3 to large groups of 20 or more—the facilitator offers opportunities for participants with disparate communication styles to find their voices and become involved. Each discussion is followed by a short period of reflection on how well the group embodied its stated values, along with making a plan to work on one of the common problems, such as how to handle varying levels of cooperation, dominance by some, side conversations, talking over one another, interruptions, lack of interest, and balanced participation; or to improve dynamics such as handling silence, showing respect, listening actively, building on each other's contributions, and asking questions.

The approach is being used in Haiti to invent a non-hierarchical educational system in a historically hierarchical society by fostering participative leadership among teachers creating and delivering curriculum for students who are thereby learning to take responsibility for their own future. The curriculum of Walden Schools uses this methodology as a tool for educating the whole person, and it has been suggested as a tool for teams comprising members from societies that stifle disagreement or encourage docility over autonomy. For these reasons, and because it has been consistently used successfully, this type of discussion-based community development has a natural place in the bottega.

Pedagogy

Given the radical reformations implicit in the bottega environment and the project-based curriculum, it is not unwarranted to expect that the behaviors and roles of faculty and students will also be significantly different from those in conventional approaches.

In a community of learning, as in a bottega, individuals have differential levels of mastery as well as of specialization. These distinctions do not, however, translate into exclusivity—only to degree of ability. For example, teaching is not exclusively the province of masters, nor is learning strictly the province of students. Anyone may teach by sharing expertise, however constrained, with someone lacking that expertise. Responsibility for learning actually increases as one gains mastery, exceeding the expectations placed on novices, because the latter necessarily focus their learning on immediate and satisfiable chunks.

Similarly, questions about *X* need not be submitted only to the *X*-specialist nor is sole authority to answer vested in the *X*-specialist. Anyone may attempt to answer any question to the best of their ability. Because the studio is an open environment, answers may be monitored by those with greater depth in *X* giving rise to some unusual benefits:

- the understanding of the answerer is open to evaluation in terms of how well he or she heard the question, connected to the appropriate answer, and expressed the answer understandably to the questioner
- the extent to which an understanding of *X* has been diffused throughout the community is made evident
- discussion of questions/answers provides an opportunity for discovering new metaphors, cross-disciplinary links, and transmission vehicles.

Except for absolute novices (and even in that instance a case can be made for peer-to-peer mentoring) everyone in the bottega may find themselves engaged in filling the role of mentor. So what is that role?

Mentors

The mentor role has two aspects, preparation and delivery. Preparation includes: designing experiences; preparing expository materials, and, defining connections. Delivery includes: monitoring, guiding, and evaluating.

In its simplest form, *designing experiences* is the design and construction of projects for student engagement. For a project like building a student advisor, this would require specifying what is

to be accomplished, identifying tasks that must be completed to meet the specifications, outlining the skills to be gained by completing the exercise, and preparing specific expository items that supply knowledge presumed in the skill. The collective of mentors have the responsibility to assure that a set of problems is available that ensure coverage of—guarantee an encounter with—all the bits of knowledge required to produce graduates anywhere in the three-dimensional space discussed earlier.

Preparing expository materials presumes a "just-in-time learning" context; i.e., they are offered in response to a demand rather than based on a pre-planned sequence of "lectures." This means you cannot make assumptions about what the listener already knows. Each expository item must carry its own context to the extent necessary to make the point central to the purpose of the unit.

Expository units might be as small and focused as a 5-minute explanation of loop control structures and benefits offered by each variant—for example, offered in response to a student noticing that some code has `do-while` syntax and another has `do-until`. The largest expository units that would be encountered in the bottega would be something like a day-long discussion of what and how regarding the use of a relational database. (Remember that units on set theory behind the concept of a relational database would be held in reserve until it is needed.) Agile approaches to development projects include the idea of a "spike"—an unplanned, discovered need for the team (or subset thereof) to be informed in some way or to experience an exploration of a less than well-understood aspect of their overall problem.

Longer expository units aimed at introducing vocabulary and basic concepts for students seeking level 1 competency are possible, but they would usually be Web-based (along with an objective exam) and available primarily for self-directed study.

Defining connections involves identifying potential opportunities for learning of the sort: "if you would like to know in more depth about," "some related information to," and, "the *X*-ologists have something to contribute to an understanding of this issue." Such opportunities include specific links/references to the resources implied by the opportunity. These may be expository materials, nested experiences, or human specialists.

Monitoring involves listening and observation. You are watching people work paying particular attention to areas where they appear to be struggling or, conversely, excelling in an interesting way. In either case the mentor is seeking to understand the essence of what is happening and address it, either by helping the individuals involved solve their problem or, in the case of success, by helping the individuals recognize their achievement and share it with others.

Guiding involves the judicious use of Socratic questioning and effective story telling. The point is to avoid telling or instructing and instead guide using leading questions or providing examples in the form of stories. Much of what a mentor does vis-à-vis stories can be informed by the ideas of Roger Schank.

Evaluating has a component focused on the learner and another on the mentor. Students are evaluated with five means:

- *objective testing*—only at the lowest level of competency, vocabulary, and foundational concepts
- *mentor and peer evaluations*—admittedly subjective, but based on experiences actually working with the evaluated individual. In the case of visiting mentors, some kind of normalization to counter individual variances will be required.
- a more objective means to evaluate students is to *compare the ration of opportunities for learning offered to them and the number actually followed*; and total followed to total mastered.
- *portfolio*—a collection of production-quality work completed by the student or by teams that included the student. Work would require approvals by mentors as appropriate and faculty prior to inclusion. The portfolio is judged both on quality of individual works and on comprehensiveness
- a novel means of evaluating students would derive from the existence of *a private social network* site that enabled mentors, especially visiting mentors, to follow the progress of individual students and assist those students with professional and/or academic placement upon graduation. The willingness of the mentor to permit a student to connect to that mentor in the network would be another, somewhat indirect, way to evaluate the performance of the student.

Mentors would be expected to be versed in the use of *pedagogical patterns* [ref] appropriate for implementing the described role. Understanding this set of patterns would improve the performance of a mentor.

In addition to standard means of evaluating mentors the additional ration of learning opportunities created to opportunities offered and of opportunities created to those followed would provide a measure of the mentor's own ability to create/innovate plus their ability to communicate their enthusiasm and the value of perpetual learning to their students.

First Steps

An attempt to approximate the ideal set forth in this paper was made at New Mexico Highlands University in 2004. After one year of operation the program was terminated for reasons having nothing to do with the value, success, or feasibility of the program.

A subsequent, less radical, program is underway at the College of Santa Fe. It is less radical in that there is more of the typical academic structure present (courses, terms, credits, etc.) than there was at Highlands. Within these restrictions, 40% of the core requirements for both the computer science and software design degrees offered consists of studios— studios that embody the ideals and pedagogy described above.

Evolution

There are more than enough faculty and professionals available to mentor students in the College of Santa Fe program and to grow that program to as many as 100 students. Within two years—as we are able to empirically demonstrate the efficacy of a program designed and delivered consistent with the ideas set forth above—we will establish a program for faculty wanting to learn by doing and the replicate our program at other schools. Semester and year-long opportunities will be available.

Plans are in place to expand the undergraduate program into a five-year program culminating in a M.S. degree and to create a

curriculum that can be shared with high school students as enrichment and/or advanced placement opportunities.

Transformation

The ultimate success for this program would be the establishment of a recognized community with a distinct culture (values, world-view, behaviors) and shared experiences, members of which demonstrate advanced capabilities and talents. In a sense, school *qua* school will disappear and merge into the overall communities in the penumbra of our field with learning being the background against which all professional and personal endeavors are engaged. In this way, the bottega is like Alexander's Gate [ref], which itself is a metaphor for his pattern languages; the Gate, once entered, is absorbed into the mind and soul of the architect and disappears.

Membership in this community will come to be the ultimate "certification"—an endorsement grounded in demonstrable (and demonstrated) achievements coupled with the imprimatur of a master mentor with a lineage of master-master transmission, within a particular philosophical school—much like the model followed in martial arts and in the fine arts.

Arms Wide Open

We expect, based on experience with the Highlands experiment, that the bottega approach to learning will be well suited to a variety of learning styles, and hence will be more congenial to women and under-represented groups. The bottega naturally gravitates toward being noncompetitive (in style, not results), and this perhaps will make a difference to the diversity of successful students.

Can We Do It?

Some of the ideas here might seem radical and disturbing: completing an entire undergraduate education in what amounts to a 1-room school, just-in-time fractional units rather than a thoroughly planned curriculum, using projects in a technical major to vector off learning into nonscientific disciplines. Moreover, the skill level required of the mentors seems off the charts.

But this approach is more like the way older, British universities were organized, and before that the bottegas of the Renaissance. Our current situation does not create excellent craftspeople to design and build the software of the future, and our researchers seem to be entranced with small steps at the frontiers. The approach we propose recognizes the reality of where we are in the discipline of computer science—more of an art than an engineering discipline, more in the infancy of its theoretical underpinnings than mature.

Mentoring does not require a heroic effort, but the effort of a team, including members of the entire faculty of the school who collaborate to uncover and satisfy opportunities for learning. And as the approach matures, its graduates are possibly natural candidates to be mentors.

Moreover, the techniques for teaching fine arts students—doing while paying attention and critically reflecting—can serve us well in our budding engineering and scientific discipline.

The experiment is worth doing.

References

On the way.