Environments for Creativity – A Lab for Making Things

Ellen Yi-Luen Do
Georgia Institute of Technology
Atlanta GA 30332-0155 USA
ellendo@cc.gatech.edu

Mark D Gross
Carnegie Mellon University
Pittsburgh, PA 15213 USA
mdgross@cmu.edu

ABSTRACT
We have, with our students, engaged in cross-disciplinary research in design. We describe parameters and principles that we have found helpful in organizing and conducting this kind of work. A variety of projects that have been developed in our group illustrate these parameters and principles. Our group focuses on making and we have come to see creativity as grounded in the ability to make things.

Author Keywords
Design studio, play instinct, objects to think with, rapid prototyping

ACM Classification Keywords
H5.m Information interfaces and presentation: Miscellaneous (Collaboration)

INTRODUCTION
Everyone can be creative, because everyone has the ability to create or make things.

Current interest in creativity stems at least in part from the realization that the traditional models of professional education may fall short in the changing economic context. Simply producing the most technically skilled mechanical or software engineers or architects no longer seems a sufficient strategy. A new kind of comprehensive education seems called for. Richard Florida in *The Rise of the Creative Class* [9], talks about "the three Ts" - talent, technology and tolerance. He observes that the trend of the economy and recipe for successful business is not big manufacturing, but instead a new type of knowledge-based, creative companies. These companies attract and retain smart people that bring "talents" to the table, invest in innovative "technology," and welcome new people, ideas, and cultural diversity ("tolerance"). What are the ingredients of academic environments that can attract and nurture this kind of creativity? We have been working for a number of years at various universities to create this kind of environment. Here we reflect on this experience and some of the challenges, factors, tradeoffs that we have encountered.

C&C'07, June 13-15, 2007, Washington, DC, USA.

Making Things
In keeping with our own disciplinary background — we were educated as architects and now teach in schools of architecture (at least notionally a creative field) — we begin with making things. Consider the Oxford English Dictionary's definitions of two words:

To create:

1. *Said of the divine agent: To bring into being, cause to exist; esp. to produce where nothing was before, 'to form out of nothing'*
2. *To make, form, constitute, or bring into legal existence (an institution, condition, action, mental product, or form, not existing before).*

To make:

1. *To produce (a material thing) by combination of parts, or by giving a certain form to a portion of matter, to manufacture; to construct, assemble, frame, fashion.*
2. *Of God (also of Nature personified, etc.): to create (a material or spiritual object).*

"Create" is a word of Latin derivation, and "Make" is a northern one, but both mean much the same thing. Create has stronger divine connotations, whereas make is humbler[1].

Creativity, in other words, is simply the propensity or ability to make things. The things may be physical, such as jewelry or bicycles, or they may be things that have no material presence, such as songs, poems, or software. These domains seem radically diverse, but based on our experience in making different kinds of things we have come to believe that there are strong commonalities between being creative in different domains. However, it is difficult to see these commonalities unless one has experience with making in at least two different domains.

We see creativity not as an innate ability, but as a capacity that can be cultivated through experience making things. Making can be learned, largely through practice, which is the dominant mode in schools of art and design. In our

[1] Use Visual Wordnet to compare "create" and "make" for an interesting perspective on their noble and humble connotations:
http://kylescholz.com/projects/wordnet/

design computing studio-laboratories we have emphasized a strategy of exploration by making things.

Materials and processes

If, as we argue, creativity is bound up in making things, then we must look at how people learn to make things, and how they learn to make things well. A look at a design school curriculum, the Bauhaus foundation courses for an historical example, or any contemporary school of architecture or industrial design, reveals an emphasis on materials and process.

A potter must know clays and glazes and the various processes by which clays and glazes are prepared, formed, and fired. A clothing designer must know fabrics and fasteners, and the various processes for sizing, cutting, and sewing.

The need to know materials and processes holds not only for traditional domains of making. The same applies to software and software-intensive systems. A programmer must know hardware and software and the processes by which code can be designed, written, debugged, and maintained. Knowledge of materials and processes — obtained through direct experience — is fundamental to the ability to make things in any domain.

Space and the Studio-Laboratory

We have worked primarily, though not exclusively, with students who are studying or have studied architectural design: undergraduates, professional master students, and PhD students in design computing. These students (and indeed all students in design disciplines) are accustomed to the studio model of learning and practice, described by Don Schön in *The Design Studio* [39]. (Schön's work on what he aptly named "the reflective practitioner" expands on his observations of architectural education in a Mellon Foundation study [38].) The design studio and the research laboratory both depend on a common space in which work takes place and is visible for informal discussion and open critique. A shared space for work is, we believe, a basic ingredient of a creative community. Although this observation may seem obvious to those familiar with this pattern, it is not a universal model in the university. For example, the lab model is virtually unknown in the humanities [15].

Below, we review some projects that we have worked on over the past few years. We do this to reflect on the creative communities we have fostered at universities where we have worked. We have never explicitly described our research practice as a curiosity-based designer-as-maker approach, but we engage problem solving and problem seeking to encourage people to *see no boundaries* between fields. We encourage exploration by constructing (interface, interactions, software, and hardware) as a process that creates, in Seymour Papert's phrase, "objects to think with" [32]. This approach is embedded deeply in the design studio culture. We set up an environment to encourage and nurture creative mindsets and approaches. Specifically, we encourage the process of generating ideas and building prototypes through incremental refinement.

All design involves a developmental process. The design ideas and eventually the artifacts that stand for the ideas (the prototypes) move from one developmental stage to the next. The process is driven by the conditions of the environment. Usually, a project remains in a particular stage until some other conditions happen to push us to move it to the next stage. At other times we don't seem to be moving to the next stage, but later we realize that we were in an incubation stage that absorbs and responds to the changes and integrates them. So what are the "driving forces" that we engage in our creative laboratory practice?

The Creative Team versus the Leonardo Model

One successful model for creative communities is to foster team-building among people who have different abilities, and some have studied how such team-building can happen [29]. For example, an artist and a programmer might work together on the design of a game. The artist is not expected to be a programmer, and the programmer is not expected to function as an artist. Rather, each member of the "creative team" functions as an expert in his or her domain; and the team learns to function together effectively by dividing responsibilities according to expertise.

Rather we favor what one might call the Leonardo model. We encourage individuals to transgress traditional disciplinary boundaries and learn to function in whatever fields of knowledge they need to accomplish their goals. In this model an artist who has an idea for a game would simply build the game, learning (or having already learned) to program along the way.

True, most people find mastering even one discipline to be a serious challenge, and only an occasional outlier will master two or more disciplines. Not to understate the work involved, but with motivation and access to knowledge, designers can acquire skills to function effectively in two or more quite different fields. Indeed, for some, knowledge of designing within one discipline can support rapid and sophisticated acquisition of knowledge in another.

If working across disciplinary boundaries holds creative riches, it is also certainly not without its challenges. In an article on the emerging field of computational biology, Junhyong Kim [23] makes trenchant observations on the nature and challenges of interdisciplinary research collaboration:

> *"While combining the knowledge of two different fields can be difficult, we can overcome such problems if we work hard and do our homework. There is absolutely no reason why an expert in biological sciences should not also be deeply knowledgeable in computer science, mathematics, and statistics…..*
>
> *Specifically, when two experts get together, they expect each other to stay within their own domains and communicate solely through some narrowly prescribed interface.*
>
> *… To make interdisciplinary research successful, we must jettison this idea of the expert. All knowledge is equal. Indeed, if we really knew*

which knowledge is important and which is not, we could all use it with shared certainty. Growth of knowledge, whether personal or fieldwide, is haphazard and full of windings and intricate turnings."

Everyone cannot know everything; tradeoffs must be made. Yet there is value in learning how "other" disciplines work, not just from the perspective of how to collaborate with others, but to understand and see designing from within more than one domain.

Hill-finding and Hill-climbing

We aim to build prototypes that extend the dimensions of a design space, rather than optimize within existing design space dimensions. This makes it difficult to make useful comparisons with other designs that serve the same function or perform a similar task. On the other hand, without evaluation it is difficult to judge the quality of the work that has been done.

One of our students, Gabe Johnson, put it like this;

One perspective is that when we build a novel tool, we will have some idea about how it might be used and how it could help, but not well enough to form a detailed evaluation plan before building it. We have to build it before we know which questions are appropriate to ask and evaluate. This isn't hill climbing, this is hill finding.

An opposing perspective holds that we shouldn't build tools without having a prescient knowledge of how that tool fits into the landscape of existing tools, and exactly what specific benefit we believe we can derive from that tool. This is the standard scientific approach of hypothesis - experiment - analysis (repeat). In other words, this is hill climbing.

PROJECTS

We organize the projects below into three categories to discuss the process and dimensions of the projects in our creative communities. All are about "making things", building computationally enhanced artifacts that are objects to think with, to play with, to contemplate ideas about design. Three patterns of promoting creative engagements emerge: (1) owning the problem, (2) design and the play instinct, and (3) building tools to make things.

Related Work (Projects)

We are well aware of much related work for the various student projects. For example, systems similar to Gesture Modeling include Surface Drawing [37] and Pinch Glove [26]. Related to the Immersive Redliner, early work on annotation in virtual reality is reported in [3]. Telepresence has been extensively explored by Tang and Minneman [44] and others [45]. Tangible music toys are numerous, for example Sony's Block Jam; see also [16] and the work of Eisenberg, Resnick and colleagues [8, 34]. Like Easigami, Ju's Origami Desk [19] supported origami learning with a physical interface. Related tangible storytelling work include work by Druin et al. [31]; there are indeed conferences devoted to interactive storytelling technology [11]. Pen based sketching systems to create 3D models, similar in intent to the Furniture Factory include work of Lipson and colleagues [30]; see also [6, 13, 18, 21, 25]. Tangible building blocks projects related to roBlocks include the early work of Aish [1] and more recently that of Anderson and Marks et al. at MERL and Watanabe's work on ActiveCubes [2, 46].

Owning the Problem or Deciding What to Design

In a traditional architectural design studio, work begins with a clearly defined problem statement, or "program" (e.g., a community library, a house for a working couple, or a train station). This way of working is appropriate where the goal is to teach and learn specific skills that every architect must know, such as arranging functions in a floor plan or deciding on a structural system to support the building. Nor do we mean to belittle the importance of learning to design things for others. The drawback—from the more general perspective of learning to make things—is that being given a ready-made problem avoids the framing question of "deciding what to design" [41]. Importantly, also many find it more difficult to take ownership of a problem that someone else has prepared.

Therefore we encourage our students to define their own problem statements—figuring out the "wants". For example, one might be frustrated with existing technology or practice and have a want for something better. Or, these wants can come from personal experience, the desire to live a smarter, more efficient, or happier life. Having wants ensures that there is a desire or passion for something to happen. This motivates people to engage in just-in-time learning to achieve their project goals. The process begins with the egocentric ("I want … ") and moves toward to a more shared vision of the benefits of a project ("we get …"). As Buchanan [5] points out, the old design education focuses on "teaching the materials, tools, and techniques of design as the primary subject matter," the new course "focuses on projects and problems that are situated within the experience and motivation of students." He argues that- "having a reason to design gives focus and purpose to student development. When a purpose exists, we find it easier then to introduce materials, tools, and techniques."

Our "own the problem" approach might seem unorthodox—at least among colleagues in Human-Computer Interaction and Design—in that we do not begin with a user-centered approach, conducting ethnographic studies, cultural probes, or other means to identify and understand the dimensions of a problem to be solved. Rather, we draw on personal experience and personal needs as a primary source for creative exploration into the design space.

Many innovations come about because the inventor solves a problem for him or herself. Stallman, for example, built emacs because he wanted a better text editor [43]. He knew the problem well — it was his problem —so he did not need to conduct surveys or observations to understand the client or the context.

Paul Graham puts it like this:

You're most likely to get good design if the intended users include the designer himself. When you design something for a group that doesn't include you, it tends to be for people you consider to be less sophisticated than you, not more sophisticated.

That's a problem, because looking down on the user, however benevolently, seems inevitably to corrupt the designer. I suspect that very few housing projects in the US were designed by architects who expected to live in them. [12]

Graham is certainly right about the architects.

Gesture Modeling

Our Gesture Modeling project [22] began with a frustration with using WIMP interfaces to create architectural form. ("I want to gesture and shape spaces"). Ariel Kemp, an architecture graduate student with a bachelor degree in computer science, wanted instead to use his hands to generate three-dimensional form, to design with computers as freely as one could in making sculpture out of clay. Ariel had previously written some image processing code that he used in the Gesture Modeling project to recognize different hand gestures, and connected this code to a 3-D geometry engine and linked the gestures to different form-making and editing operations.

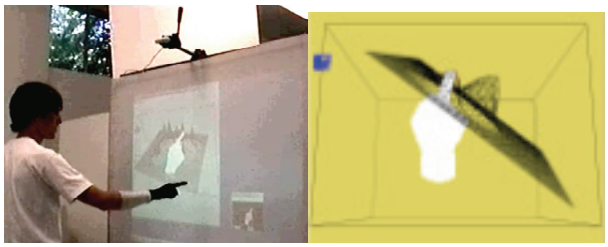

Figure 1. Deforming a mesh model with a hand gesture.

Immersive Redliner

The Immersive Redliner project [20] came from the need to collaborate with other stakeholders on design artifacts. (I need a way for my clients to give feedback). Thomas Jung, an architecture graduate student, understood that need. He felt that designing in the medium of CAD tended to isolate, rather than bring together, the various stakeholders, and he wanted to support a conversation among stakeholders about the artifact being designed. In order to realize this goal he learned to program in Java and mastered the intricacies of the Java3D graphics environment. The Redliner software offered stakeholders a desktop virtual reality model in which they could browse a 3D model on the Web and post annotations about particular features on objects in the design.

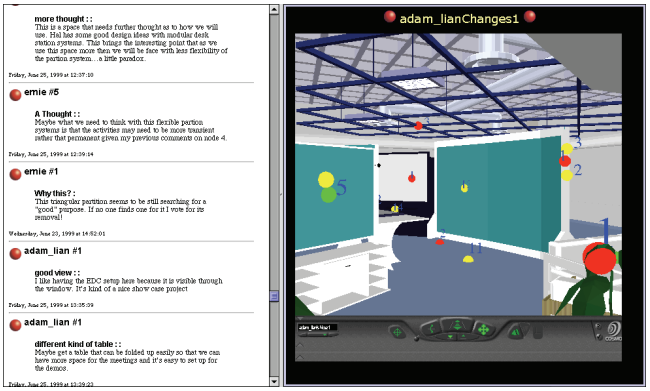

Figure 2. Redliner Annotations by stakeholders in an interior design

Telepresence Tables

Telepresence Tables are an instance of calm technology that provides one person an ambient awareness of others while protecting privacy. ("I want to stay connected with my friends and family while maintaining my privacy.") Originally a project in a "Home of the Future" class we taught, the Telepresence Tables are two small tables outfitted with electronics so that shadow patterns made on one table appear in colored light on the other. Each table has an 8x8 array of light sensors and two colors of LEDs that project upward to make patterns on a frosted plastic surface. A microcontroller in each table collects the light sensor values, echoes them locally by lighting yellow LEDs, and sends the data to the other table, which displays the remote shadow pattern by lighting the red LEDs. People find the light patterns beautiful and the experience engaging.

Figure 3. Each Telepresence Table translates shadow patterns into light and transmits them to the other table.

The Telepresence Tables were built over two months by a team of four: Ken Camarata and Mike Weller, two PhD students in computational design, Kursat Ozenc, a PhD student in interaction design, and Bridget Lewis, an undergraduate physics major. The team started by brainstorming around the idea of ambient awareness with privacy, and moved rapidly to tangible interaction embedded in furniture. The team developed several alternatives quickly, and selected the grid of photosensors and LEDs. Over the course of the first few weeks the team together worked out the electronics and software design,

learning basic circuit design and fabrication as they went. A shared workspace and a (persistent) whiteboard supported their design discussions and served as informal communication between team members. The design underwent continual refinement as the team built small prototypes to test various aspects. The team made up for their collective lack of experience in analog electronics by opportunistically taking advantage of resources. For example, rather than send boards out to be fabricated (which no one on the team knew how to do) they opted for using a computer-numerically controlled mill to mill circuit paths in copper plated boards.

Tinkering, design, and the play instinct

The second pattern in our projects is the importance of play. The late American graphic designer Paul Rand described designing as a kind of play within given or self-imposed constraints [33]. Play – an exploration of materials and processes — is what distinguishes routine acts of making, that is to say production, from more creative acts of making that may result in innovative ideas.

Papert [32] used the French word *bricolage,* or what computer scientists and artificial intelligence researchers called "hacking" goes to the heart of what creative people do, and people who aspire to being creative must practice.

This sort of creative play is encouraged in schools of design (which include architecture, industrial, communication and interaction design) and the arts (including music, painting, sculpture, and drama). Students learn to make things by making things.

Despite some advantages, hacking, tinkering, and playful exploration are often disparaged. These activities are seen as not sufficiently goal-oriented: A good engineer, it is said, begins with a clearly articulated problem statement, and then applies reliable methods to reach a solution. A student who spends time playing around with things is wasting time that could be more profitably spent applying known methods to the problem at hand. And inevitably there will be failures — as actually building a prototype reveals unanticipated behaviors that undermine a previously plausible design idea. Getting the plan right in the first place would avoid wasted time and costly mistakes.

The difficulty with this position is, of course, that creative work, particularly design work, seldom begins with a clearly stated problem. Rather, as many have pointed out [35, 39, 42], design is as much concerned with identifying and expressing a problem as it is finding solutions. Adages such as "Defining the problem is the problem" and "The problem and solution co-evolve" exemplify this well-known characteristic of designing.

Bach Blocks – playing with music

Shaun Moon's Bach Blocks are a set of colored blocks, a camera, and some software that reads the arrangement of blocks and plays a tune. Shaun, a graduate student with a background in architectural design, wanted to play with music, and wanted in particular to build a toy that could engage his young children with music.

Colors represent pitches and the positions of the blocks determine the sequence of play. Thus, Bach Blocks is at once an instrument for making music and a notation to compose it. Ordinarily the software plays from left to right (and blocks arranged vertically play harmonies), but Bach Blocks software can also be set to play the tune in any direction.

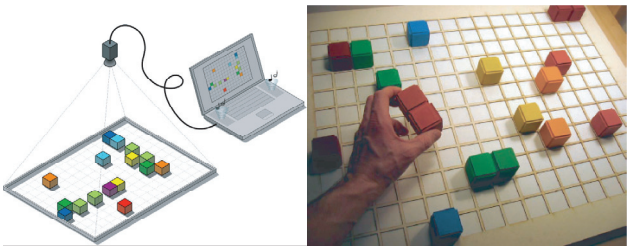

Figure 4 Bach Blocks: making and playing music with colored blocks

Easigami – playing with origami

Playing with origami, children learn geometry and spatial reasoning skills. However, children often find it difficult to interpret diagrams in a book into origami action. The traditional way of teaching origami discourages children from creating original paper models. It cannot reveal the rich content in the transformation between a 3D model and its 2D crease pattern. Easigami is a tangible user interface that addresses these issues. It uses computer interaction to clarify origami actions and to encourage origami exploration through 2D-3D transformation.

Easigami's physical interface is a paper-like triangle tile toy composed of flat triangle pieces and electrically enhanced hinges. The hinges join plastic triangles to form a flat sheet with a crease pattern. Each hinge can provide folding instructions by illuminating LEDs to indicate which crease(s) are active as well as the direction to fold. Each hinge senses the relationship between the two adjacent triangles it connects, and sends the angular information to a desktop computer. Users can follow instructional signals and fold the Easigami interface along pre-folded creases. A real-time computer graphic model of the physical triangles is displayed on screen along with the corresponding 2D crease pattern.

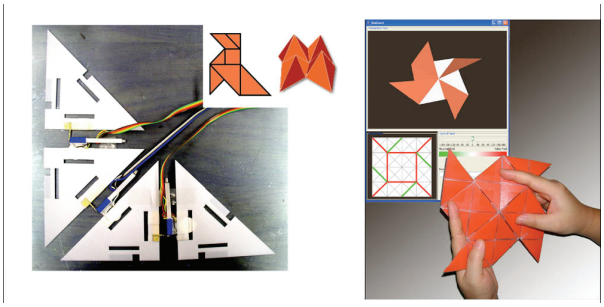

Figure 5. Easigami senses a user's folding pattern.

Easigami was built by Yingdan Huang, a PhD student who had an undergraduate degree in architecture and had taken advanced courses in programming and computer graphics. Easigami was her first tangible media project, and she had no experience with wiring up analog electronics or microcontroller programming. However, she was able to quickly produce alternative physical designs for embedding electronic components in a physical model, and develop a hierarchy browser that enables Easigami users to see where the current state of the folded sheet is in the space of possible origami designs. She was also able to quickly build an Open-GL 3-D viewer that dynamically displays the model as the user folds it, although she was stymied for some time by the realization that accurately modeling paper folding, notably the transient states in which paper is flexed into complex curved surfaces, is a non-trivial problem. With some effort we were able to convince Yingdan that an approximate simulation would be an adequate representation.

Storytelling Cubes – building animations tangibly

Storytelling Cubes are a tangible device for young children to create animated stories on a computer screen. Each cube contains an orientation sensor (three mercury switches) and a wireless transmitter that tells the host computer which side of each cube is facing up. The faces of cubes depict various characters, scenes, and actions in an animated cartoon. Children use the Storytelling Cubes to illustrate their spoken stories in a create-your-own-adventure fashion. Tony Sheng-Kai Tang, an architecture PhD student, interfaced the Storytelling Cubes with the Alice graphical programming environment developed by Randy Pausch's group at CMU to enable children to use the Storytelling Cubes to generate animated cartoon stories.

Figure 6. Making animations with Storytelling Cubes

Building Tools to Make Things

The third pattern in our projects is a tool-building approach. We are less interested in making particular designs for a particular client or user, than in developing ways of working — methods and tools— that can open up new design spaces. We mention briefly two projects that illustrate that approach: Furniture Factory and roBlocks.

Furniture Factory -- Sketch to fabrication

The Furniture Factory program helps designers make physical prototypes using rapid prototyping and manufacturing machines. It provides a sketch-based design interface that a designer can use to draw furniture in 3-D. The program then displays the model in an isometric viewing window where the designer can view it and edit it. It then decomposes the 3-D model into flat panels and displays them in the parts window. Furniture Factory adds joints where one panel connects to another according to connection conditions. These added joints enable designers to construct the physical model easily and quickly. The program generates HPGL code to cut the furniture parts on a laser cutter.

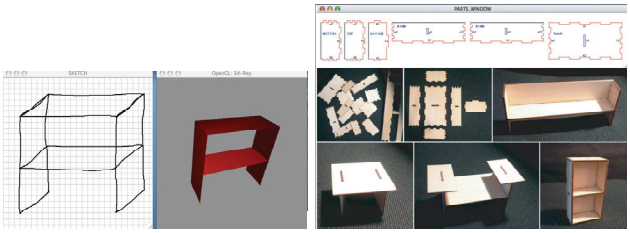

Figure 7. Sketch to Fabrication with the Furniture Factory

Furniture Factory is designed specifically to produce a simple subset of the universe of things that can be made from flat material. We are working on a generalization of the program that will allow a designer to sketch and manufacture a wider variety of things, based on a larger language of form that includes folding, laminating, assembling, and cutting flat materials.

roBlocks – a Robot Construction Kit

The roBlocks construction kit [40], built by Eric Schweikardt, a PhD student with an undergraduate major in architecture and a minor in computer science, is a set of 40 mm plastic cubes that snap together using small neodymium magnets in their faces. Each block contains a microprocessor and custom circuit boards that are glued with conductive epoxy to the face magnets. Power, as well as data, is transmitted from one block to the next through the magnets and a small spring pins also mounted on each face. Some blocks are input blocks: they have light, sound, touch, or other sensors. Others output: they have motors, lights, or speakers. Still others are logic or arithmetic blocks that combine signals from the sensor blocks.

The idea of roBlocks is that designers who have no previous experience with electronics, mechanical design, or programming can assemble the components of a robot by making a block construction. The configuration itself also programs the robot's behavior, so (conceptually), a light sensor block snapped to a motor block will be a robot that moves toward light; adding a NOT block between them produces a robot that avoids light.

Figure 8. roBlocks construction kit for modular robotics

DISCUSSION

For various reasons the kind of environment we foster—an interdisciplinary, no-boundaries, technically sophisticated studio-laboratory—is still unusual in the university. Certainly we are not alone: Pelle Ehn articulates a similar vision for a Digital Bauhaus at Malmö University [7]; Stanford's "d-school", the ID-StudioLab at Delft, the MIT Media Lab, and others are ventures in the same vein. Within the design disciplines, and especially industrial and interaction design there is a growing interest in hybrid models of education [24, 28, 36]. Still we find that the model runs against the grain of the university, which tends to reward focus within, rather than across, disciplines.

We recognize the irony in our emphasis on making as a means to creativity just as Western economies are less and less about making things. Can we learn creativity through making, yet apply it to other (non-making) domains?

Education of Designers vs. that of Engineers

Architects and designers are educated in a quite different fashion than scientists and engineers. Architects in particular are integrators and therefore negotiators among a set of diverse other experts. Like designers in other disciplines, architects are taught to keep options open, explore parallel alternatives, celebrate ambiguity. Engineers tend to be more goal-oriented and stay within their field of expertise and treat ambiguity as something to be eliminated.

Thus we find that it is easier for design students to learn technical skills (programming, electronics) they need to carry out projects in design computing than it is to teach engineering and computer science students to work in ill-defined situations.

Engineering and computer science students tend to be less well prepared for open-ended investigation than those who have studied design. Engineering and computer science students with whom we have worked are happiest when we present them with a specification of work to be accomplished. An open-ended brief tends to make them uncomfortable—they do not know where to begin or how to proceed. Once we give them a clear objective they can apply the skills that they have acquired to attain it. Design students, in contrast, even when given a clear specification, will ignore it and do something else!

Although it is possible that becoming expert in technical domains is simply easier than learning to design, we suppose that other factors are responsible for this phenomenon. One is cultural arrogance: due to a perceived 'pecking order' in the university, computer science and engineering students do not recognize that there is an important skill that they do not know. Indeed, their education is intended to prepare them to walk up to any new problem and apply their bag of tricks to it.

In our experience a computer science student or an engineering student is more likely to jump from an initial statement of a problem to propose a method of solving it, and then immediately pursue that approach without considering alternatives. Generating and comparing alternatives is an activity that is drummed into designers throughout their education.

There is also the matter of being able to accomplish something. It is usually possible for a student of computer science or engineering who knows how to write code to make *something*—however poorly designed or inelegant—without having first learned the skills of designing. On the other hand, a design student who sets out to make an artifact that has software or hardware must — perforce — learn some technical skills.

Of course, there are problems going the other way as well. Not every designer finds computation a natural medium. Students from a design background are sometimes overwhelmed by the amount of technical detail that they must master in order to do anything interesting. Those who start to learn to program by taking courses in the computer science department are often bored by the examples used in problem sets (which have no relationship to anything they might be interested in) and on the other hand surprised by the precision that is demanded to make a working piece of software. Other students view programming as a way to get something done — "by any means necessary," and failing to recognize the power of good design in software, produce horrible kludges that work (perhaps) for a key example or two but in the end limit any further exploration that can be done with the prototype.

Designing and Programming

If creativity is about making things, and making things is about design, what is the place of programming in all this? Many who have worked both as a programmer and in some other domain of design recognize powerful parallels.

Consider the words of master programmer Dick Gabriel in "the Poetry of Programming." In 2004 Gabriel received the AAAI/ACM Allen Newell Award "for innovations not only on fundamental issues in programming languages and software design but also on the interaction between computer science and other disciplines, notably architecture and poetry." In an interview titled "The Poetry of Programming", Gabriel said:

> *Writing code certainly feels very similar to writing poetry. When I'm writing poetry, it feels like the center of my thinking is in a particular place, and when I'm writing code the center of my thinking feels in the same kind of place. It's the same kind of concentration. So, I'm thinking up possibilities, I'm thinking about, well, so how do I reinvent the code, gee,*

you know, what's the simplest way to do this. [10]

Or Paul Graham again: "Hacking and painting have a lot in common. In fact, of all the different types of people I've known, hackers and painters are among the most alike. What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things."[12]

de.sign = pro.gram

Martin Brynskov at the Center for Interactive Spaces at the University of Aarhus [4] pointed out that the words design and program are remarkably close in their Greek and Latin roots. According to the Oxford English Dictionary the word "design" is made from the prefix "de-" (meaning out) and the root "sign" (meaning mark) that is, to mark out. Likewise, the word "program" is made from the prefix "pro-" (meaning forward or out), and 'gram" (meaning writing). That is, both design and program mean to mark out or make an explicit representation.

Emphasis on Prototypes

We emphasize the value of building working prototypes of ideas—quickly and focusing on the parts of an idea that are interesting or that seem worth exploring. In traditional design education, prototypes are physical models that illustrate the form, and perhaps the materials, of the design to be made. As computation is added to the mix, it becomes more difficult for traditionally trained designers to make working prototypes that illustrate not only the physical and material characteristics of designs but also their functional behavior. Software environments like Flash are frequently used to mock up interaction, but Flash is a poor language to build or even model more sophisticated computational behavior. As designers grapple with making things that have both physical and computational characteristics, the need becomes apparent for prototyping tools that can capture and convey not only the superficial aspects of a design, but also carry the more fundamental ones. Work on toolkits and design environments for physically embedded computation—as in [14, 17, 27] can make this kind of work far more accessible.

The New Makers

If as Richard Florida and others argue, creativity is crucial in the new economy, then perhaps we can foster creativity by putting making back into education. There is nothing new about that idea, but for a variety of reasons, some outlined in Dick Buchanan's "Design and the New Learning" [5] learning to make things has become conspicuously absent in most courses of higher education. One might expect schools of engineering to teach students to make things, but engineering curricula are strong on teaching analysis and principles and light on the actual practice of making. Certainly, making things is still taught in two places in the university: schools of design and the arts, and departments of computer science. Surprisingly, as the field of computer science matures, the skills of making software are being systematically displaced by more analytic skills. In short, as it matures as a discipline, computer science seems to be moving away from teaching design, just as other engineering disciplines such as mechanical or civil engineering did in their earlier days.

Now is an interesting moment. Things have changed, and the ways of making things have changed too. Almost everything in our world is the product of design. Increasingly designing is mediated by computational processes. Increasingly the artifacts that we encounter —our shoes, our houses, even our parks, are embedded with microcontrollers, sensors, and electronics. Designers of the future—the New Makers—will need to be fluent with the materials and processes of computation, in addition to the materials and processes of other domains.

We believe that powerful insights are available to those designers—who come initially from whatever discipline—who master the art and craft of making things in more than one domain. These insights may eventually further the development of what Simon termed a "Science of Design" [42]. Meanwhile, we hold that creativity is rooted in the experience of making things.

ACKNOWLEDGMENTS

We appreciate the work of our anonymous reviewers in suggesting revisions for this paper. This research was supported in part by the National Science Foundation under Grant ITR-0326054 and the Pennsylvania Infrastructure Technology Alliance. The views and findings contained in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

1. Aish, R. 3D Input for CAAD Systems. *Computer-Aided Design*, *11* (2). 66-70.
2. Anderson, D., Frankel, J., Marks, J., Leigh, D., Ryall, K., Sullivan, E., Yedidia, J,. Building Virtual Structures with Physical Blocks. in *UIST 1999*, ACM, 1999, 71-72.
3. Bolter, J.H., L.F.; Meyer, T.; Nichols, A. Integrating perceptual and symbolic information in VR. *Computer Graphics and Applications, IEEE*, *15* (4). 8 - 11.
4. Brynskov, M. pro.gram = de.sign, Personal Communication, Pittsburgh, 2006.
5. Buchanan, R. Design Research and the New Learning. *Design Issues*, *17* (4). 3-23.
6. Davis, R. Sketch Understanding in design: Overview of Work at the MIT AI lab. in Davis, R., Landay, J. and Stahovich, T.F. eds. *Sketch Understanding, Papers from the 2002 AAAI Symposium*, American Association for Artificial Intelligence (AAAI), Menlo Park, CA, 2002, 24-31.
7. Ehn, P. A manifesto for a Digital Bauhaus. *Digital Creativity*, *9* (4). 207-217.
8. Eisenberg, M., Eisenberg, A., Gross, M., Kaowthumrong, K., Lee, N. and Lovet, W. Computationally-Enhanced Construction Kits for Children: Prototype and Principles. in *Proc. Int'l Conf. Learning Sciences*, Seattle, WA, 2002, 79-85.

9. Florida, R. *The Rise of the Creative Class: And How It's Transforming Work, Leisure, Community and Everyday Life*. Basic Books, New York, 2003.
10. Gabriel, R. The Poetry of Programming, SUN Microsystems, 2002 (accessed Oct 16, 2006). http://java.sun.com/features/2002/11/gabriel_qa.html
11. Göbel, S., Spierling, U. and Hoffmann, A. (eds.). *Technologies for Interactive Digital Storytelling and Entertainment: Second International Conference, TIDSE 2004 Proceedings* Springer (Lecture Notes in Computer Science), Darmstadt, Germany, 2004.
12. Graham, P. *Hackers and Painters*. O'Reilly, Sebastopol, CA, 2004.
13. Gross, M.D. and Do, E.Y.-L. Drawing on the Back of an Envelope: a framework for interacting with application programs by freehand drawing. *Computers and Graphics*, *24* (6). 835-849.
14. Hartmann, B., Klemmer, S.R., Bernstein, M., Abdulla, L., Burr, B., Robinson-Mosher, A. and Gee, J. Reflective physical prototyping through integrated design, test, and analysis. in *Proceedings of the 19th annual ACM symposium on User Interface Software and Technology*, ACM Press, Montreux, Switzerland, 2006, 299-308.
15. Hiatt, G. We Need Humanities Labs *Inside Higher Education*, 2005, (accessed Oct 26, 2005). http://www.insidehighered.com/views/2005/10/26/hiatt
16. Hornecker, E. Tangible Interaction Wiki - Music Applications, 2007 (accessed Jan 2007). http://www.media.tuwien.ac.at/e.hornecker/Tangibles.html#Music_Applications
17. Hudson, S.E. and Mankoff, J. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. in *Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM Press, Montreux, Switzerland, 2006, 289-298.
18. Igarashi, T., Matsuoka, S. and Tanaka, H. Teddy: A Sketching Interface for 3D Freeform Design. in *Proceedings SIGGRAPH 1999 Annual Conference on Computer Graphics*, ACM, 1999, 409-416.
19. Ju, W., Bonanni, L., Fletcher, R., Hurwit, R., Judd, T., Post, R., Matthew Reynolds and Yoon, J. Origami Desk – Integrating Technological Innovation and Human-Centric Design. in *Proceedings Conference on Designing Interactive Systems (DIS)*, 2001, 399 – 405.
20. Jung, T., Do, E.Y.-L. and Gross, M.D. Sketching Annotations in 3D on the Web. in *ACM Conference on Human Factors (SIGCHI)*, ACM Press, Minneapolis, 2002, 618-619.
21. Kara, L.B. and Stahovich, T.F. Pens & sketching: Hierarchical parsing and recognition of hand-sketched diagrams. in *Proceedings of the 17th annual ACM symposium on User interface software and technology UIST '04*, ACM, 2004, 13-22.
22. Kemp, A. and Gross, M.D., Gesture Modelling: Using Video to Capture Freehand Modeling Commands. in *Computer Aided Architectural Design Futures 2001*, (Eindhoven, NL, 2001), Kluwer, 33-46.
23. Kim, J. Computers Are from Mars, Organisms Are from Venus. *IEEE Computer*. 25-32.
24. Klemmer, S.R., Verplank, B. and Ju, W., Teaching embodied interaction design practice. in *Proceedings of the 2005 conference on Designing for User eXperience*, (San Francisco, California, 2005), AIGA: American Institute of Graphic Arts, Article 26.
25. Landay, J.A. and Myers, B.A. Sketching Interfaces: Toward More Human Interface Design. *IEEE Computer*, *34* (3). 56-64.
26. LaViola, J. and Zeleznik, R., Flex and Pinch: A Case Study of Whole Hand Input Design for Virtual Environment Interaction. in *Proceedings of the Second IASTED International Conference on Computer Graphics and Imaging*, (1999), 221-225.
27. Lee, J.C., Avrahami, D., Hudson, S.E., Forlizzi, J., Dietz, P.H. and Leigh, D., The calder toolkit: wired and wireless components for rapidly prototyping interactive devices. in *Designing interactive systems: processes, practices, methods, and techniques*, (2004), ACM, 167 - 175
28. Lundgren, S., Torgersson, O., Hallnäs, L., Eriksson, E. and Ljungstrand, P., Teaching Interaction Design: Matters, Materials and Means. in *DRS Wonderground conference*, (2006).
29. Mamykina, L., Candy, L. and Edmonds, E. Collaborative creativity. *Comm. ACM*, *45* (10). 96-99.
30. Masry, M., Kang, D.J. and Lipson, H. A Pen-Based Freehand Sketching Interface for Progressive Construction of 3D Objects. *Journal of Computers and Graphics*, *29* (Special Issue on Pen-Based User Interfaces). 563-575.
31. Montemayor, J., Druin, A., Farber, A., Simms, S., Churaman, W. and D'Amour, A. Physical Programming: Designing Tools For Children To Create Physical Interactive Environments. in *Proceedings of the Conference on Human Computer Interaction*, 2002, 299-305.
32. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York, 1980.
33. Rand, P. Design and the Play Instinct. in Kepes, G. ed. *Education of Vision* Braziller, New York, 1965, 154-173.
34. Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K. and Silverman, B., Digital manipulatives: new toys to think with. in *SIGCHI conference on Human factors in computing systems*, (Los Angeles, California, United States, 1998), 281-287.
35. Rittel, H. and Webber, M.M. Dilemmas in a General Theory of Planning. *Policy Sciences*, *4* (1973). 155-169.
36. Sato, K. and Verplank, W. Panel: teaching tangible interaction design. in *Proceedings of the conference on Designing Interactive Systems: processes, practices, methods, and techniques*, ACM Press, New York City, New York, United States, 2000, 444-445.

37. Schkolne, S., Pruett, M. and Schröder, P. Surface drawing: creating organic 3D shapes with the hand and tangible tools. in *Conference on Human Factors in Computing Systems (CHI '01)*, ACM, Seattle, 2001, 261-268.
38. Schön, D.A. Learning a Language, Learning to Design. in Porter, W.L. and Kilbridge, M. eds. *Architecture Education Study*, Andrew W. Mellon Foundation. New York, 1981, 339 - 471.
39. Schön, D.A. *The Design Studio*. RIBA, London, 1985.
40. Schweikardt, E. and Gross, M.D. roBlocks: A Robotic Construction Kit for Mathematics and Science Education. in *International Conference on Multimodal Interaction*, ACM, Banff, Alberta, 2006, 72-75.
41. Shaw, M., Herbsleb, J. and Ozkaya, I., Deciding what to design: closing a gap in software engineering education. in *Proceedings of the 27th international conference on Software engineering (ICSE)*, (2005), ACM, 607-608.
42. Simon, H. *Sciences of the Artificial*. MIT Press, Cambridge, MA, 1969.
43. Stallman, R. My Lisp Experiences and the Development of GNU Emacs, 2002. http://www.gnu.org/gnu/rms-lisp.html
44. Tang, J.C. and Minneman, S.L., VideoWhiteboard: Video Shadows to Support Remote Collaboration. in *CHI '91*, (New Orleans, LA, 1991), ACM Press / Addison Wesley, 315 - 322.
45. Tollmar, K. and Persson, J. Understanding remote presence. in *Proceedings Second Nordic Conference on Human-computer interaction*, 2002, 41-50.
46. Watanabe, R., Itoh, Y., Asai, M., Kitamura, Y., Kishino, F. and Kikuchi, H. The Soul of ActiveCube - Implementing a Flexible, Multimodal, Three-Dimensional Spatial Tangible Interface. in *Proc. of ACM SIGCHI International Conference on Advanced Computer Entertainment Technology ACE 2004*, 2004, 173-180.