Learning When Less is More: "Bootstrapping" Undergraduate Programmers as Coordination Designers

Sirong Lin, Deborah Tatar, Steve Harrison
Dept. of Computer Science
Virginia Tech
660 McBryde Hall, MC 0106
silin@vt.edu, tatar@vt.edu, sHarrison@vt.edu

Jeremy Roschelle, Charles Patton
Center for Technology in Learning
SRI International
333 Ravenswood Ave.
<first name.last name>@sri.com

ABSTRACT
In this paper, we describe an undergraduate computer science class in the United States that we started with the intention of creating a participatory design experience to create distributed mobile collaborative technologies for education. The case highlights the ways in which programmer understanding of an innovative new technology can depend on understanding the context of use. The students were to use Tuple-spaces, a language for coordination. However, it soon became clear that while the coordination of machines may be thought of as a computer science problem, the students could not understand the technical system without richer models of how, why, or when coordination is desirable. We were in the ironic position of teaching human coordination at the same time as describing the technical properties of a system to support it. To "bootstrap" the learning process, we asked the students to draw on their own coordination expertise by implementing familiar coordinative games. We propose games as an addition to the PD toolkit when implementers need help in stepping outside their everyday mindset.

Author Keywords
Guides, instructions, author's kit, conference publications.

ACM Classification Keywords
H5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION
A person might think that teaching design to a population of largely white and Asian male computer science undergraduate students in a university context in America is an area without wonder to the participatory design (PD) community. As dePaola said in PDC 2004 "PD can bring together field-study methods and participative design activities to facilitate the creation of a common language between designers and users. The goal is to integrate 'systemic analysis, appreciative intervention, and practitioner participation'[5] to create social-technical-political conditions that reduce the gap between design practices and users' work practices"[3]. Computer scientists and their technical knowledge are in the background from this perspective. Computer scientists are thought to be well-tuned to understanding their own mastery-based culture. However, the rise of ubiquitous and pervasive computing can create situations that turn the usual engineering "object-world" [1] on its head. One such situation is discussed here, in which the professor's growing understanding of the social-cognitive barriers to student understanding of the system led to significant course redesign. Although we initially intended the students to engage in participatory design using the system, instead we were forced to create methods for addressing pre-conditions that would permit such design.

PRIOR PD WORK ON COORDINATION
In two prior projects, we had used PD with teachers to create a number of successful classroom activities on mobile wirelessly connected handheld computers for math and science learning [6, 9, 10]. These activities turned out to involve a high degree of interpersonal and machine coordination both in the foreground---the reasons one might adopt and use a system---and the background---the affordances which render the system easy and attractive enough to utilize in a group situation. As with many *classroom response systems* [7], our activities focused on increasing student learning and involvement by a) breaking out of a lecture format to support small group work, and/or b) giving the teachers more information about the student experience with the aim of influencing their instructional practice. Remarkable results included high 8th grade performance on items from the Advanced Placement Calculus exam. That is, students at the beginning of the algebra learning sequence were able to understand calculus concepts taught only peripherally in the course of a one month intervention.

Figure 1 illustrates an example of a foreground coordination activity, Match-My-Graph. In this activity,

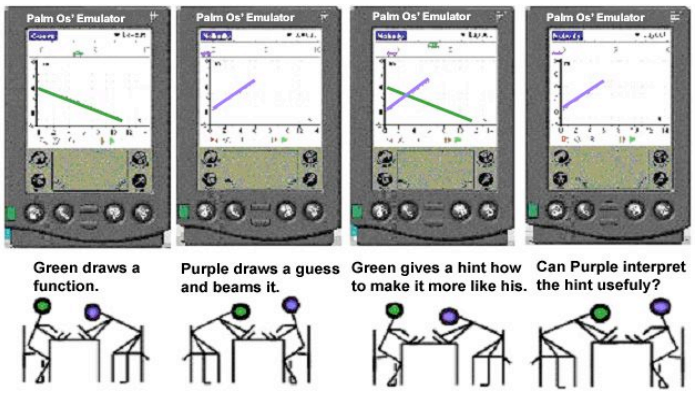

Figure 1: An activity that involves a high degree of coordination between pairs of learners working face-to-face (Reprinted from Tatar, 2003).

pairs of algebra learners learned to use math language better by resolving the differences between their screen displays. One student, the grapher, would create a graph of a function. The other student, the matcher, had the job of creating the same graph, based on verbal hints from the grapher. After the matcher heard the grapher's hint, he/she would send his new guess to the grapher. The grapher would then look at the two graphs, the original and the guess, to formulate a new (verbal) hint.

Note that in this activity, the primary site of learning is in the student finding the precise words for verbal expression of the math concept. The computer merely creates an impetus and setting for doing so.

Background coordination considerations include the mechanisms by which students are grouped into pairs, how the correct work is distributed to the students (and handhelds), and how students attain the roles of matcher or grapher [10]. In all of these cases, we observed and teachers reported high degrees of a) responsibility allocated to the students for controlling their own behavior and b) fluidity in goals. For example, one teacher planned on having the students work in mixed ability groups on a given day. She started the class by assigning students to pairs, but when some of the children were called out of the room, her priorities changed, and she broke off from a discussion, called out "Find someone to work with," and continued her discussion.

In response to this, we design for situated action [8] with flexible goals and roles. For example, from a machine point of view, students became pairs *by acting like* they were a pair, that is, by communicating with one another via machine. One student took on the role of a matcher *by acting like* a matcher, that is, by sending his/her first guess to the person who, by accepting it, became a grapher.

TUPLE SPACES

Tuple spaces originated as a mechanism for exploring different paradigms for allocating work to asynchronous, parallel-distributed computers [2]. The underlying support for coordination was that: a) A tuple *client* would break a problem up into different components (tuples) posted or *written* to the Tuple Space server. b) Different tuple clients could *take* components that matched their specifications and execute them. c) Clients would decide whether to take one or more tuples by a process of associative (template) matching. That is, a student looking for a partner, might find one by asking the tuple server to match a specific name (like "Deborah") or a more general one (anyone in the group "RedDiamond"). Tuples are well suited for changing coordination. The absence of need for the programmer to attend to details of machine connectivity (such as socket numbers) or complex heavy-weight data structures (as in database programming) reduces complexity. Additionally, they emphasize light-weight, emergent mapping of machine to activity, of person to machine, and of person/machine to role.

THE CLASS

Fourteen senior and first-year graduate students at Virginia Tech participated in a project-based class to build tuple-space based handheld applications for education. Thirteen were male, one female. Four were white, one African-American, eight were Asian and one Asian-American. Students received and signed informed consent letters. A separation was maintained between their grade in the class and their participation in the research. Pre and post class interviews were conducted, all work was collected for the research and classes were video and audiotaped. Programs were implemented in Java, using T-Spaces and Swing or SWT for laptops or Dell Axim X50 Handheld Computers.

COMPUTER SCIENCE STUDENT OBJECT-WORLD

Our initial approach to teaching the class was to present the underlying technology interleaved with examples of the kinds of programs we had developed with teachers and implemented successfully in the past. We were surprised by the lack of questions about key elements. When we asked the CS students to engage in an initial design exercise planning a simple, Tuple-based interaction, some were unable to proceed. Others proceeded but turned in designs that, for example, created a database system inside Tuple spaces, with information assurance, normalization and security features.

Our initial response was to treat this as a purely technological misunderstanding. We created a template-matching exercise in which we used paper to walk the class through the communication of information between tuple clients and servers. This clarified a certain component of the process for the CS students, but they expressed dissatisfaction with what we thought of as the attractive simplicity of writing and taking.

We came to see that the problem was at the level of the CS student object-world:

> ….it is the object as they see and work with it that patterns their thought and practice, not just when they must engage the physics of the device but throughout the entire design process, permeating all exchange and

> discourse within the subculture of the firm. This way of thinking is so prevalent within contemporary design that I have given it a label—"object-world" thinking... [1], p. 4

The technical teaching problem was tied to two intertwined topics: the invisibility of human coordination activities and a disciplinary focus on control, efficiency, and correctness.

IMPLEMENTING GAMES TO TEACH COORDINATION

To help the CS students focus more clearly on a setting of deliberate coordination between multiple parties, we asked them to create coordinated, collaborative versions of games in Tuples spaces. Seven games were created, of which six were variants of popular and/or educational games: collaborative crossword puzzles, hangman, "Apples-to-Apples"™, "Krypto"™-Telephone, math bingo, and "Pictionary"™. Additionally, a seventh game was created, the algorithm enactor.

Each of these games reified some special challenge of motivating and implementing coordinated activity. For example, Apples-to-Apples™ is a game in which all players draw from a set of cards, each of which has a distinct noun or noun phrase on it (like "Shirley Temple," or "canasta"). One player, the judge, turns over a card with an adjective or adjectival phrase. Players play the card from their hand with the noun that they think the judge will think corresponds best to the adjective. For example, if the adjective is "curly," the player might pick "Shirley Temple" rather than "canasta." There are no right or wrong answers. The fun lies in picking the best answer from a small set of alternatives and in seeing how other people associate noun and adjective phrases. The game can be played by any number of players, but at any given time, there is one judge. The role of judge moves between players on every round, offering opportunities for rule negotiation and challenging the notion of designing for one fixed "teacher" role. Human and machine coordination is illustrated in Figure 2.

In the collaborative crossword puzzle design (Figure 3), turn taking was negotiated entirely socially. Anyone could take a clue at any time and post an answer. If one person

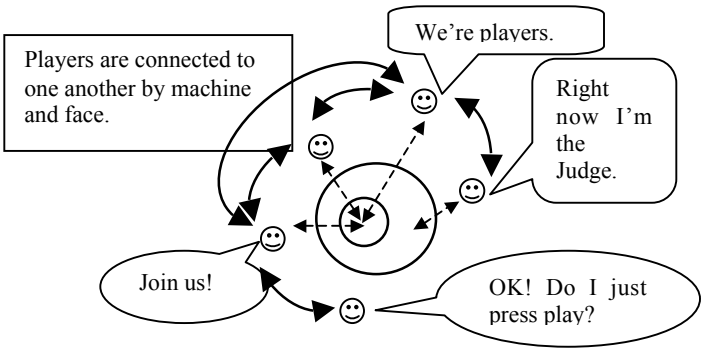

Figure 2: The coordinated, distributed Apples-to-Apples™ game emphasizes the lack of particular right or wrong answers, the role of communication outside the computer, and the need for flexible movement between roles.

posted an answer that put an "e" in a slot and another posted an answer with a "d" for that same slot, both would be displayed for discussion. Initially the students designed mechanisms for allocating points, which entailed highly complex mechanisms to prevent "cheating" by substituting one's own answers for someone else's. However, in design review, other students raised many other ways of cheating, leading to the design team to question 1) their behavioral expectations from the users and 2) the merit of giving points in a game that already constitutes its own reward.

VALUES INHERENT IN COMPUTER SCIENCE

Through interviews, in-class comments, designs and reactions to existing systems, two overlapping themes in the CS student object-world emerged: their views of the benefits of computing, and their views of desirable control.

The CS student object-world appeared to highlight four possible benefits to computing in the classroom: that computers would know the right answer, that they could store a complete record of activity, that they could forestall student mistakes, and that they could give and remember points or grades. For example, some thought that algebra students would learn better in Match-My-Graph if the system told them when the two lines were the same rather than having them negotiate/discover this. The CS students anticipated and were concerned that the algebra students might think that the two lines were the same when in fact they were not (although this never actually happened). The CS students were puzzled by the absence of features in Tuple spaces to support these putative benefits. They did not see the student process as particularly significant.

Control, in the CS view, belonged to the centralized server, as it does in most web programs, which would allocate jobs and roles to student-users, and record and evaluate behavioral history. The CS students felt that it was a deficiency in the program that it did not automatically monitor the algebra students to report how many rounds of Match-My-Graph were successfully completed, which seemed to be a self-evident metric of interest to any teacher. A better system was a system in which the teacher would personally authorize the giving out of material at prespecified time intervals and ensure that everyone was working on equivalent problems at the same time. It, further, seemed obvious to the CS students that every participant should be identified by name, and that steps had to be taken to make sure that the name given was correct and canonical.

These views of the benefits of computing and of control had curiously strong parallel with their views of what a good class entailed (and therefore what participatory design might look like and of the kinds of information they should seek to find from participating teachers).

The game context appeared to loosen the CS view of appropriate and desirable behavior and allow a level of questioning not permitted in the discussion of "serious" topics.

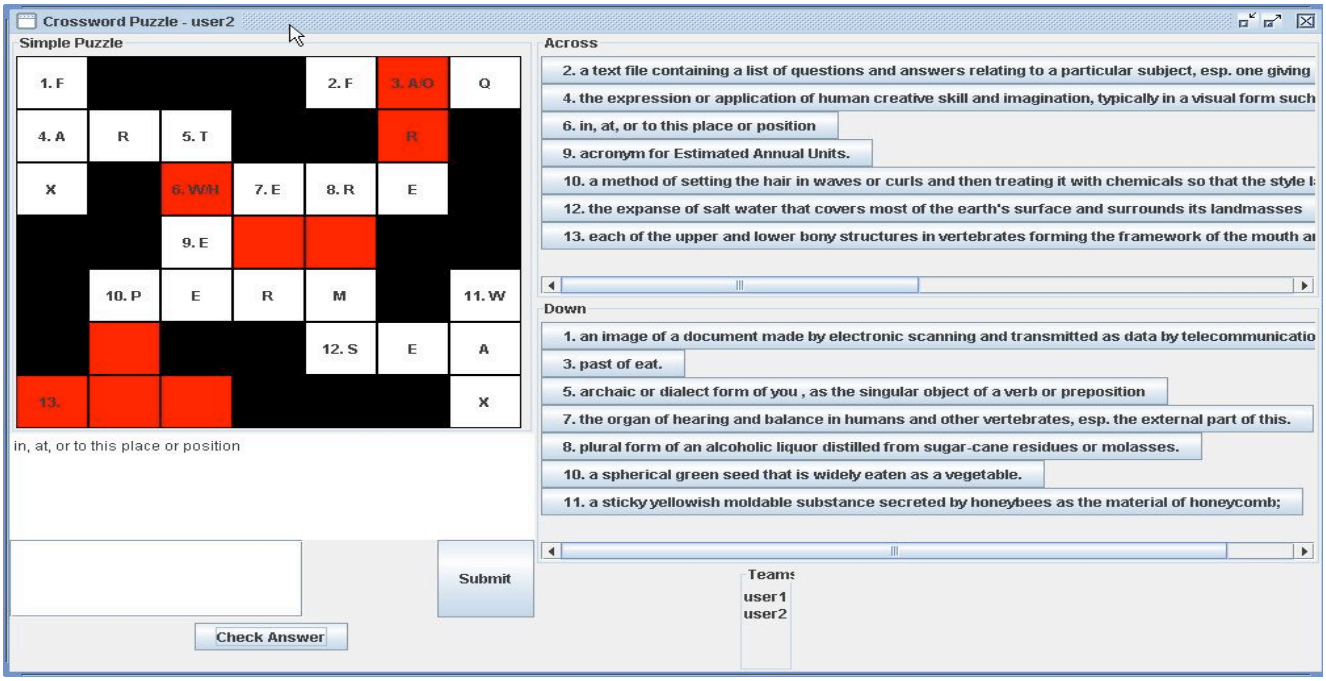

Figure 3: The collaborative crossword puzzle game simply displayed conflict, leaving the resolution to the social world.

DISCUSSION

As it happened, the CS students were interested in games, and the games motivated them to think more deeply about coordination, motivating their understanding of system features. The success of this activity is in part a result of their object-world; because they were computer science students, they took the computer-based implementation of games as a self-evident desirable goal. Others might have asked "Why do you want to complicate with technology a game like Hangman that is perfectly adequate with paper and pencil, or chalkboard and chalk?" They did not.

The usual PD toolkit includes participatory activities like mockups and joint storyboarding. Other recent work has used games as a bridge between designers and future users [4]. We document a case in which these tools failed because of a strongly held engineering worldview. We are using games quite differently, to make certain elements of the social world more visible through reification and analogy. Because the deep design of our technical system originated in participatory design experiences, it embodied technical features that were challenging for this worldview, though part of it.

Thus the experience reported here is not precisely participatory design, nor is it systems design. However, it a pre-requisite for both if the world of ubiquitous and pervasive computing is to embody situated, emergent, flexible, human-scale systems. In this case, games proved to be a good context for opening the minds of engineers to alternative social meanings of computing. We propose games as an addition to the PD toolkit when implementers need help in stepping outside their everyday mindset.

ACKNOWLEDGEMENTS

This work was supported by NSF ITR/IERI Grant #REC-REC 0427783. Thanks to all the members of the Tuples project, CS4984 Fall 2005, and PDC reviewers.

REFERENCES

1. Bucciarelli, L. *Designing Engineers*. MIT Press, 1996.
2. Carriero, N. & Gelertner, D. *How To Write Parallel Programs: A first course.* MIT Press, Cambridge, MA, 1990.
3. dePaula, R., Lost in Translation: A Critical Analysis of Actors, Artifacts, Agendas, and Arenas in Participatory Design. In *Proceedings of the eighth conference on Participatory design* (Toronto, Ontario, Canada, 2004), ACM Press, 162-172.
4. Donovan, J. & Brereton, M., Meaning in Movement: A Gestural Design Game. In *Proceedings of the eighth conference on Participatory design* (Toronto, Ontario, Canada, 2004), ACM Press,
5. Karasti, H. (2001) Bridging Work Practice and System Design: Integrating Systemic Analysis, Appreciative Intervention and Practitioner Participation. *Journal of Computer Supported Cooperative Work*, 10 (2). 211-246.
6. Penuel, W. & Yarnall, L. (2005) Designing Handheld Software to Support Classroom Assessment: Analysis of Conditions for Teacher Adoption. *Journal of Technology, Learning and Assessment,* 3 (5).
7. Roschelle, J., Penuel, W.R. & Abrahamson, A.L. (in press) Using classroom networks to improve achievement and participation in mathematics and science. *Educational Leadership.*
8. Suchman, L.A. *Plans and situated actions.* Cambridge University Press, Cambridge, 1987.
9. Tatar, D., Roschelle, J., Vahey, P. & Penuel, W.R. (2004) Handhelds Go to School. *IEEE Computer*, 36 (9). 30-37.
10. Vahey, P., Tatar, D. & Roschelle, J., Leveraging Handhelds to Increase Student Learning: Engaging Middle School Students with the Mathematics of Change. In *ICLS* May, 2004).