Proselytizing Pervasive Computing Education: A Strategy and Approach Influenced by Human-Computer Interaction

D. Scott McCrickard and C. M. Chewar
Center for Human-Computer Interaction and Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061-0106
{mccricks, cchewar}@cs.vt.edu

Abstract

A course on pervasive computing should be structured around key functions throughout a systems development process to cover common underlying concerns throughout science and engineering disciplines—development of design rationale, prototyping, evaluation, and component reuse. However, broader considerations of usage context and appreciation for research and methodological contributions from other disciplines must be strongly factored into course planning. To achieve both goals, we suggest learning objectives, a general strategy and approach using case-based learning.

1. Introduction

Pervasive computing drives many new concerns and solutions, such as new programming, architecture, and network paradigms, and requirements for design and evaluation activities. How should a pervasive computing class be developed and included in an undergraduate or graduate computer science program? How would central themes taught in this course complement other courses, like software engineering, human-computer interaction (HCI), or even theory or systems topics? What are some successful pedagogical approaches for teaching pervasive computing?

We approach these questions from the perspective of a professor at a mid-sized to small school, tasked with the job of developing a semester-long course on pervasive computing within a computer science program, though only vaguely familiar with the topic at all. Certainly, resolving questions like these is an important and promising endeavor—we already have seen firsthand the excitement and engagement it inspires in students. Judging from current evidence of groundbreaking applications that extend into many different disciplines, coupled with this technology's potential to more deeply touch the lives of ordinary people, pervasive computing will undoubted be a hot area for research contribution and industry employment in the years to come. With pervasive computing applications, understanding implications that result from shifting technical and social constraints necessitates a broad, multidisciplinary perspective in its students—but with this comes a unique ability to push the boundaries of our own disciplines. Pervasive computing certainly is a topic departments would want to include to address state-of-the-art computing.

Perhaps our professor starts to appreciate challenge of developing a pervasive computing class trying to nail down a definition of the term itself and a consensus of its subtopics. Part of the difficulty may be in identifying fields or more traditional topics in which it is rooted as a sub-discipline. Clearly, there are HCI concerns that drive the field, but most HCI textbooks and handbooks do not yet include a chapter on the area. Several new journals and conferences address many important and specific questions, but there are few seminal papers to help the computer science professor carve out the essential questions. The heavy technology implementation component of the subject is grounded in computer engineering and network concerns and surrounded by social and ethical questions, clouding disciplinary underpinnings even more. Before long, deeper challenges with connecting the many topics together to form a coherent course theme will surely confront the course developer, as will integrating the pervasive computing course theme of with other parts of the computer science program.

These three challenges—*course content selection*, *thematic integration*, and *programmatic integration*, contain many subtleties that should be factored into course planning, such as identifying critical learning objectives, approaches used in related courses, and typical aspects of computer science student culture. We explore these subtleties in section 2. Section 3 presents a strategy and pedagogical approach for addressing enduring pervasive computing learning objectives. An example demonstrates how the three challenges with our strategy and approach.

2. Key Education Challenges in PerCom

In a review of recent research within ubiquitous computing, Abowd and Mynatt note three important interaction themes pushed by the research community: natural interfaces, context-aware applications, and automated capture and access [2]. These themes broaden the reach of computational devices and provide a basis for pervasive computing. By introducing the term *everyday computing* to apply to systems providing support for continuous, daily actions executed concurrently and interrupted by other activities, Abowd and Mynatt define an essential characteristic of pervasive computing [2]. This concept is further articulated by several goals: everyday practice of people must be understood and supported, activities must be augmented with devices that support new experiences, and devices must be networked for a holistic user experience [1]. To support everyday computing and make progress toward acceptance by sizable users groups, they argue that two important research challenges within the ubiquitous computing field must be resolved—evaluating systems and solving social issues [2], as well as defining appropriate physical interaction techniques [1]. The importance of these issues in the broad development of the everyday computing initiative is a pivotal message that must transfer to coverage of pervasive computing topics.

The work Smailagic and Siewiorek incorporates a user-centered design approach to respond to development challenges of handheld devices and wearable computers [8]. The interdisciplinary design methodology they have developed consists of five stages, including a requirements analysis and walkthrough, participatory response to storyboards, mock-up prototype creation, software implementation and testing, and full implementation and field trial evaluation. A strength of this approach is that it has been tested by more than two dozen generations over a ten-year period, resulting in numerous examples of process and product. Another key benefit provided to students is the big picture appreciation of concurrent design activities throughout a variety of disciplines.

Specifically within the software development and HCI domains, there have been many recent arguments for important research directions related to pervasive computing. A recent workshop on mobile, ad-hoc collaboration argued for a comprehensive probe into increased empirical research to understand important usage opportunities, development of prototype systems and applications, and sociological research on the impact of such technology [5]. Researchers at ubiquitous computing conferences argue for activity-centered approaches to design to support diverse and distributed activities common to the field [3], with a focus on requirements development rather than merely creating and implementing new technologies.

In practice, issues related to the broader impact of technology often beleaguers its introduction and adoption—privacy concerns and ethical implications. Some argue that engineers developing new technologies must thoroughly understand all possible consequences from appropriate as well as unintended use, and design them to be impervious to malicious adaptation [9].

2.1. Pervasive computing learning objectives

The important arguments related to pervasive computing can be expressed as several pervasive computing learning objectives. Computer science and HCI students should be able to:

- Explain key enabling technologies, user and group activities and goals, and application domains that drive innovation in pervasive computing state-of-the-art.
- Appreciate a wide variety of social implications, ethical dilemmas, and policy issues that are brought about because of pervasive computing, to include concerns for privacy and liberties.
- Understand how multidisciplinary activities drive and contribute to system lifecycle development stages.
- Demonstrate how traditional computer science and HCI skills and methods can adapt to solve pervasive computing research, design, and evaluation problems.
- Recognize a variety of new solution approaches for overcoming technical issues related to pervasive computing implementation concerns.
- Identify new opportunities for pervasive computing innovation and application redevelopment, to include support for new community and collaborative activities, improving universal access issues, providing new forms of interaction and display, and resolving usability problems.

These learning objectives are deliberately selected to focus on broader themes that will remain relevant in a quickly changing technology landscape, preparing students to make long-term contributions to pervasive computing research and development. While it is tempting to select a variety of interesting and important topics to paint a loosely connected survey of a new area, this will often not provide students with as rich of an experience as a course designed around enduring themes. Unfortunately, it seems like survey approaches (such as the course described in [7]) are already being used to teach pervasive computing topics. Course designs that exercise system lifecycle development decision-making include benefits of active and situated learning, but often suffer from the time required to master specific programming languages or resolve/coordinate technical difficulties—often at the expense of bigger picture topics that transcend current implementation techniques.

3. Proposed Strategy and Approach

To teach pervasive computing to computer science and HCI students, we propose a general strategy that will achieve thematic and programmatic integration. Coupled with a case-based learning approach that allows flexible and wide ranges of course content selection, the three challenges for teaching pervasive computing are addressed.

3.1. Strategy: Reiteration for Function Focus

One possible strategy for teaching pervasive computing follows from an approach we have used in teaching HCI to computer science students. Like the emerging field of pervasive computing, technological change has certainly been a very important factor of interface development in general over the past decade, a challenge that has been confronted when considering educational approaches to HCI. Rather than presenting a variety of loosely connected topics, we have sought thematic and programmatic integration by organizing the course around several visitations of a design process. As depicted in Figure 1, in each design process iteration, we focus on a different development function and set of supporting methods, tools, and issues: developing design rationale, prototyping, evaluation, and reuse. For example, in the first phase of the course, a lifecycle design process (such as a scenario-based design method) is introduced to the students. In the next phase, all steps in the design process are revisited while considering various methods and outcomes supporting the function of developing design rationale. In the third phase, each step of the design process is again revisited, but while considering prototyping functions instead. As prototyping throughout a design process confirms and extends design rationale, we can draw parallels between contemporary approaches. In the next stage, methods, tools, and issues related to evaluation functions are introduced as approaches for prototyping and validating rationale (thus producing reusable artifacts that would be appreciated in terms of the design cycle in the final stage of the course).

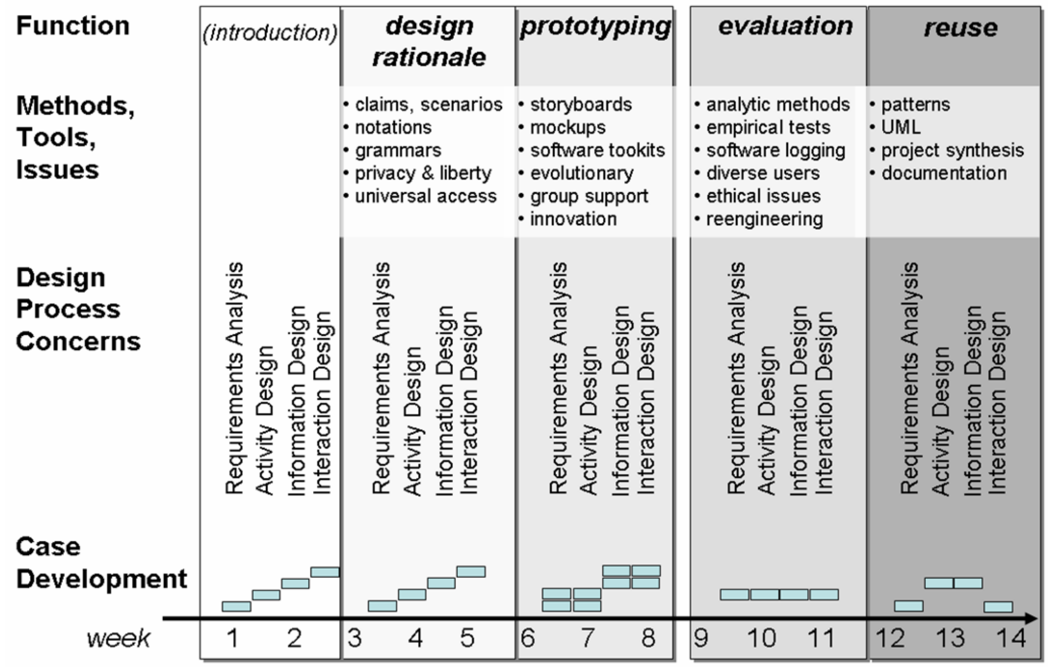
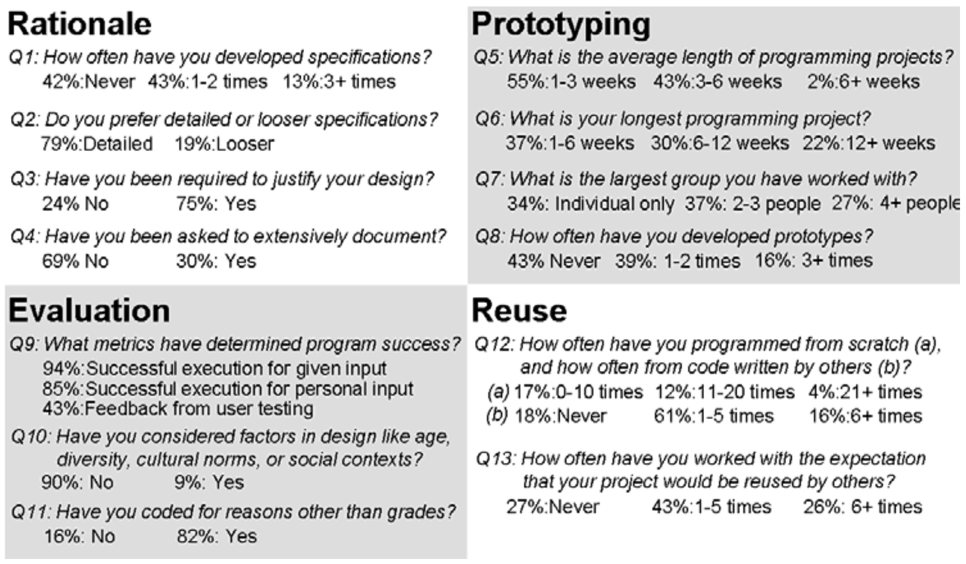


Figure 1. Reiteration for Function Focus (RF3), a strategy recommended for achieving thematic and programmatic integration in pervasive computing education.



Rationale

Q1: How often have you developed specifications?
42%:Never 43%:1-2 times 13%:3+ times

Q2: Do you prefer detailed or looser specifications?
79%:Detailed 19%:Looser

Q3: Have you been required to justify your design?
24% No 75%: Yes

Q4: Have you been asked to extensively document?
69% No 30%: Yes

Prototyping

Q5: What is the average length of programming projects?
55%:1-3 weeks 43%:3-6 weeks 2%:6+ weeks

Q6: What is your longest programming project?
37%:1-6 weeks 30%:6-12 weeks 22%:12+ weeks

Q7: What is the largest group you have worked with?
34%: Individual only 37%: 2-3 people 27%: 4+ people

Q8: How often have you developed prototypes?
43% Never 39%: 1-2 times 16%: 3+ times

Evaluation

Q9: What metrics have determined program success?
94%:Successful execution for given input
85%:Successful execution for personal input
43%:Feedback from user testing

Q10: Have you considered factors in design like age, diversity, cultural norms, or social contexts?
90%: No 9%: Yes

Q11: Have you coded for reasons other than grades?
16%: No 82%: Yes

Reuse

Q12: How often have you programmed from scratch (a), and how often from code written by others (b)?
(a) 17%:0-10 times 12%:11-20 times 4%:21+ times
(b) 18%:Never 61%:1-5 times 16%:6+ times

Q13: How often have you worked with the expectation that your project would be reused by others?
27%:Never 43%:1-5 times 26%: 6+ times

Figure 2. Student survey responses, summarizing incoming experience and attitudes of junior/senior CS and CE majors, as related to the four functions of our strategy

This course framework, which we refer to as Reiteration for Function Focus (RF3), allows students to be exposed to a wide variety of methods and design-related issues, but facilitates higher-level learning objectives that require comparison, synthesis, and abstraction. We feel that this approach would be useful for achieving the learning objectives expressed previously. Our recommendation for placing emphasis on the functions of developing design rationale, prototyping, evaluation, and reuse is based on two claims (supported by evidence in the next section):

1) Upper-level undergraduate students typically lack practical experience to appreciate the multidisciplinary role of each function within a development processes,

2) Each function represents enduring concepts firmly grounded in engineering and scientific process, yet is renewed through contemporary methodological and technological research contributions and uniquely able to address social/ethical issues and spur innovation.

3.2. Programmatic obstacles

In developing a computer science course that focuses on an emerging area, we wanted to connect individual topics on methods, tools, and issues to the wider development functions of developing design rationale, prototyping, evaluating, and reusing components. A key factor in recommending this strategy was the nature of practical experience incoming students would typically have at that point. Recognizing that students must experience some exposure to and even frustration with these functions to prime continued interest in learning, we conducted a survey of undergraduates signed up for Introduction to HCI. These students are all junior or senior year computer science or computer engineering majors, representative of the population expected in a pervasive computing class. The survey probed experience in each of the four functions; results are listed in Figure 2.

After reviewing the results of our survey, we were pleased to see some areas in which students would be primed to learn about methods, tools, and issues related to these functions. For example, we are pleased that students were often confronted with experiences beyond a grade that made them value the quality of a product they developed. It was also encouraging that students tend to have some base of experience in justifying their designs and reusing components. However, there are many other points of concern apparent in, especially related to the average lengths of project experience, thinking about issues related to universal access and user diversity, and generating program requirements. These student responses reinforce the importance of structuring presentation of emerging topics around these four functions—essential aspects of professional and academic computer science efforts. However, they also motivate the selection of a particular pedagogical approach, discussed next.

3.3. Approach: Case-based learning

To address the functions and learning objectives we propose, many which involve the examination of social issues, we sought a pedagogical approach that would bring students quickly to key decision points and allow them to sort out the alternatives, but compensate for lack of functional experience. While hands-on experience would allow practice of some of the functions and parts of the learning objectives, we fear it would be at the expense of the big picture, especially in a class on emerging technologies without robust and widely accepted tools. Instead, the case method uses short discussions and activities framed on case materials to help students learn key concepts.

Users of the case method in similar disciplines tout it as providing increased student engagement, improved analytical skills and decision-making abilities, and enhanced application of concepts to practical problems [4]. Case methods are quite common in business schools, law education, and design disciplines like architecture. While case methods are not a commonly practiced form of pedagogy in computer science, there are reports of successful uses, such as in teaching discrete event simulation, compiler design, operating systems, and computer architecture classes, allowing analysis of complex design tradeoffs. Although cases are used in slightly different ways and formats in each of these other fields, they do have common characteristics. Like any challenging HCI development project, the professional activities in each of these other disciplines require substantial time and resources—far more than can be approximated in a classroom setting. As a solution, cases provide a rich context in which to test or discuss a concept, an efficient starting point for a realistic, interactive experience. Other benefits in using cases in education are apparent as well, such as how continuous exposure to new cases helps instructors maintain awareness of professional practices [4].

Our ongoing research is exploring various material formats and classroom discussion techniques that are most suitable for HCI instruction [6], and we anticipate that many of our findings will be relevant to teaching pervasive computing as well. To more clearly illustrate our proposal for using a case-based learning approach, the next section looks at how a series of lessons could be taught in a pervasive computing class.

3.4. Integrating the strategy and approach

To demonstrate how a pervasive computing class could be taught with the use of our strategy and a case-based learning approach, we illustrate an arbitrary point in a semester, perhaps weeks 4-7 (see Figure 1). At this point, the design process concerns are revisited for the second and third times, looking at how the functions of design rationale and prototyping are accomplished through various methods and tools. While some of material would be delivered in traditional lectures and out-of-class readings, an integrated case study of an actual design effort provides depth. Perhaps four different case studies were introduced at the beginning of the semester, and are also revisited. One of the case studies focuses on a development effort for *e-textiles*, textiles with integrated electronics devices. Although introduced earlier to illustrate requirements analysis processes, the e-textiles case is revisited in week 4, 6, and 7.

In week 4, the e-textiles case helps students understand how to developing design rationale during activity design. Through reading the case, students can gain an appreciation of how claim and scenario writing assisted designers in identifying key tasks to support with e-textiles. Related class activities could involve generating new design rationale to express other ideas and interactive class discussion could focus on a decision-making process to sort out the most promising activities as the development process continues. Material provided by the case supplies fuel for discussion about broader social issues, such as support for universal access and considerations of privacy and personal liberty. Students may prepare position papers to justify decisions for the continuing development process based on further research probing these topics, using aspects of the case as a starting point. A wide variety of issues related to the topics at hand are probed without sacrificing many class resources.

In week 6, the case is revisited to show prototyping for requirements analysis, specifically by storyboarding and working with potential users. Existing case material demonstrates use of storyboards to probe requirements for other candidate activities. Again, this may be a starting point for critical analysis that can extend to activities selected by students in week 4. Students gain hands-on experience developing storyboards and discussing them with potential users, and then reflect on requirements for storyboarding particular to pervasive computing. Class discussion develops comparison skills, perhaps by evaluating different storyboarding methods, and provides opportunity to integrate cutting-edge contributions from the research community.

In week 7, the e-textiles case illustrates the design process function of prototyping in activity design, by having potential users interact with simple mockups. New case material continues to challenge student sensitivity to broader topics. Perhaps a simple mockup of e-textile devices can be created and used in a classroom skit, allowing students to critique a method of prototyping new activities with the devices. We can discuss how mockup prototypes can be augmented for an evolutionary implementation, and begin using actual software/hardware toolkits to build and demonstrate important system features.

While this series of lessons could be supported by an ongoing case study on e-textiles, a few additional case studies on other pervasive computing topics should also be integrated to support other lessons. This will help expose students to key enabling technologies and a wide variety of solution approaches, and certainly provide interesting situations around which to weave discussion of social issues and ethical dilemmas. Creating the case material to support this type of class should be a natural extension of research efforts, but can also be accomplished with a little bit of creativity on the instructor's part. In this manner, we connect contemporary methodological and technological research contributions to firmly grounded functions within engineering and scientific processes.

4. Conclusions and Future Work

Within this paper, we have summarized challenges and proposed solutions for course content selection, thematic integration, and programmatic integration for pervasive computing in a computer science curriculum. We base our recommendations on critical objectives suggested by research within the field, articulating suggested learning objectives. To meet these learning objectives, we propose a strategy that situates contemporary pervasive computing methods, tools, and issues within a more enduring structure, a strategy implemented by our RF3 framework. As we probed the background experience and attitudes of incoming students, we recognized that this framework will fill many gaps in their ongoing education, but must be supported with an approach that allows active learning without sacrificing undue classroom time. To this end, we recommend case-based learning and illustrate our vision with a sample series of lessons connecting contemporary methodological and technological research contributions to firmly grounded functions within engineering and scientific processes.

Our ongoing research explores the use of case-based learning in human-computer interaction (HCI). We are looking at questions related to material preparation and classroom interaction techniques. For instance, we are evaluating two different approaches for integrating case-based classroom activities with semester project work. In one approach, students revise an existing case study and develop a "version 2" interface based on what they learn, challenging them to implement, test, and document innovative ideas, yet work from an existing body of knowledge. In the other approach, students are challenged with the creation of a case study (useful as instructional material in a future course) that recreates and documents design rationale, prototyping activities, and evaluation. Through this process, students perform new research and selected reengineering, but focus on analysis and opportunity identification in the status quo. Other work we are performing looks more broadly at how case studies can be created and visualized as a natural part of an interface design process. We look forward to sharing these results and discussing other approaches with the growing pervasive computing community.

5. References

[1] Abowd, G. D., Mynatt, E. D., and Rodden, T. "The Human Experience." *IEEE Pervasive Computing 1* (1), Jan-Mar 2002. IEEE Press, pp. 48-57.

[2] Abowd, G. D. and Mynatt, E. D. "Charting Past, Present, and Future Research in Ubiquitous Computing." *ACM TOCHI 7*(1), 2000, pp. 29-58.

[3] Christensen, H. B. and Bardram, J. E. "Supporting Human Activities–Exploring Activity-Centered Computing." In *Proc. of UbiComp 2002*, pp. 107-116.

[4] Herried, C. F. "Case Studies in Science: A Novel Method of Science Education." *Journal of College Science Teaching 23* (2), Feb 1994, NSTA, pp. 221-229.

[5] Korteum, G., Gellersen, H. W., and Billinghurst, M. "Mobile Ad-hoc Collaboration." In *Extended Abstracts of CHI '92*. Apr 1992, ACM Press, p. 931.

[6] McCrickard, D. S., Chewar, C. M., and Somervell, J. P. "Design, Science, and Engineering Topics—Teaching HCI with a Unified Method." In *Proc. of the Technical Symposium on Computer Science Education (SIGCSE '04)*, March 2004, ACM Press.

[7] Rudolph, L. "What I Did on My Fall Vacation—Pervasive Computing Class." *IEEE Pervasive Computing 2* (2), Apr-Jun 2003. IEEE Press, pp 100-104.

[8] Siewiorek, D. P., and Smailagic, A. "User Centered Interdisciplinary Design of Wearable Computers." In *Human Computer Interaction Handbook*, Jacko, J. A. and Sears, A., eds. Lawrence Erlbaum, 2003, pp. 635-655.

[9] Stone, A. "The Dark Side of Pervasive Computing." *IEEE Pervasive Computing 2* (1), Jan-Mar 2003, IEEE Press, pp 4-8.