PerEd'09

Proceedings of the Second Workshop on Pervasive Computing Education

September 30, 2009

Orlando, Florida USA

Held in conjunction with Ubicomp 2009

The Eleventh International Conference on Ubiquitous Computing

William G. Griswold, Editor

Organizers

William G. Griswold
Computer Science and Engineering
UC San Diego
La Jolla, CA 92093-0404 USA

Scott F. Midkiff
Bradley Department of Electrical and Computer Engineering
Virginia Tech
Blacksburg, VA 24061 USA

Gregory Abowd
School of Interactive Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0760 USA

Program Committee

Gillian Hayes	UC Irvine, USA
John Krumm	Microsoft Research Redmond, USA
Hiroyuki Morikawa	University of Tokyo, Japan
Shwetak Patel	University of Washington, USA
Aaron Quigley	University College Dublin, Ireland

Table of Contents

A Contextual Learning Game for Toddlers Installed on an Interactive Display Attached to a Shopping Cart 1
Gerrit Kahl, Karin Leichtenstern*, Johannes Schöning, Lübomira Spassova, Antonio Krüger
*DFKI GmbH (Germany) and *University of Augsburg (Germany)*

Ubibot - Prototyping Infrastructure for Mobile Context-Aware Computing 9
Erwin Vedar, W. Brian Evans, William G. Griswold
UC San Diego (USA)

Teaching Smart Environments and Co-operative Ensembles 17
Sebastian Bader, Thomas Kirste, Christoph Burghardt
University of Rostock (Germany)

Introducing TU100 "My Digital Life": Ubiquitous computing in a distance learning environment 26
Mike Richards, John Woodthorpe
The Open University (UK)

Public Digital Note-Taking in Lectures 37
Roshni Malani, William G. Griswold, Beth Simon
UC San Diego (USA)

A Contextual Learning Game for Toddlers Installed on an Interactive Display Attached to a Shopping Cart

Gerrit Kahl
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken Germany
gerrit.kahl@dfki.de

Lübomira Spassova
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken Germany
spassova@dfki.de

Karin Leichtenstern
Multimedia Concepts and Applications
University of Augsburg
86159 Augsburg, Germany
leichtenstern@informatik.uni-augsburg.de

Antonio Krüger
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken Germany
krueger@dfki.de

Johannes Schöning
DFKI GmbH
Stuhlsatzenhausweg 3
66123 Saarbrücken Germany
johannes.schoening@dfki.de

UbiComp 2009, Sep 30 – Oct 3, 2009, Orlando, FL, USA

Abstract

Bored toddlers (children at the age of 1-3) often cause stress for parents during shopping trips in supermarkets. Sitting in the front of the shopping cart, they often grouch or arrogate different articles such as sweets or toys. The reason for this behavior is often the lack of useful activities for kids during shopping of their parents. In this paper, a concept for contextual learning games is introduced by using an interactive display attached to the shopping cart's handle bar. With this game, we want to let toddlers participate in the shopping process to a certain degree without annoying their parents. Using RFID technology, the shopping carts are able to detect the articles and products inside. These items are reflected in the game played by the toddlers. We are interested in up to which extent the integration of real world items in the game can provide a meaningful learning experience and also the needed distraction from sweets or toys. As a result, we expect parents to be more relaxed while their children pursue a useful experience.

Introduction & Related Work

Shopping with toddlers, especially young toddlers can sometimes be irritating and very stressful for their parents. In the case of parents and toddlers shopping together, particularly in a supermarket setting, many parents often have to deal with situations in which the toddlers want to a have a special product such as sweets or toys [5]. It is interesting to see that according to Ebster et al. the influence children wield over their parents' purchase decisions at the point of sale is often grossly underestimated by their parents [5]. Lindstrom [10] estimates that children at the age between 8 and 14 years spend and cause about approximately $1.2 trillion worth of sales worldwide (note that our target group is rather younger, so the volume toddlers “spend” will be smaller). In the HCI community as well as in the Ubicomp community, lots of research has been conducted to investigate the effects of advertisement on young children [6] as well as on the interaction with small interactive (mobile) displays. Brand and Greenberg [2] investigated the effect of advertisement in a classroom and Adler et al. [1] present a good overview on that research field. Ebster [5] suggests that the best way of keeping toddlers quite is to distract them from the shopping process. While this can be done by simply showing a colorful comic movie on the display of a shopping cart, we are interested in designing games for interactive displays attached to the shopping cart handle bar that let toddlers still participate in the shopping process to a certain degree without requiring too much attention by their parents. An example of such a display attached to a shopping cart can be seen in figure 1. We try to lead the toddler’s attention between the real world and some games with real world items on the interactive shopping cart display. While lots of research in the

Figure 1: The interactive shopping card attached to the handle bar of the intelligent shopping car.

Ubicomp domain was conducted on how software and hardware for kids differ from software and hardware fro adults, the interaction of toddlers with small (7 inch) interactive displays has not been considered so far. Most research done up to date has focused on the design of classical educational software [6] and not on how toddlers can be involved in learning games in the context of real world objects and activities. The Parc Tabs [14] were among the first interactive small

displays introduced by Want et al. Following the idea of interactive, mobile displays the Tuister [3] or the Display Cube [8] were some of the first projects investigating embedded displays in real world objects. The work of Leichtenstern et al. [9] explored the role of the Display Cube in social learning software and the integration of the Display Cube as interaction and presentation device. Similar to the display system attached to our shopping cart are the stationary office door displays of the Hermes system by Cheverst [4]. Recently, Merrill et al. presented the Siftables [11], which are small interactive computers with a display, speaker, wireless communication and motion sensors.

The remainder of the paper is structured as follows: First, the scenario is described in section 2. Next, we describe and discuss the design challenges when developing interactive games for toddlers. Section 4 presents our idea of a contextual learning game. Finally, section 5 provides an overview of the used hardware followed by some concluding remarks.

Scenario

Susan Meyer is a 32-years-old bank clerk. She is a loving wife and mother of a 2-year-old boy named Sam. On ordinary days, Susan has a lot of stress at her workplace. After seven hours of hard work, she picks up Sam from the day nursery. This is the time when Susan's actual stress often starts because she has to deal with the daily tasks of a housewife such as shopping. In these situations, Sam can become a tease. He is often tired and bored and starts grouching. Sometimes Sam even starts crying or doing nonsense. Then, Susan wishes to find ways to overcame this stress. With our concept, we can help Susan to cope with such situations. The instrumented shopping cart detects all goods Susan has inserted. This information can be used as input for different contextual learning games, which are displayed on the cart's touch-sensitive screen. Now, using our concept, Susan can enter the store and place Sam in the toddler's seat of the instrumented shopping cart. Then, she can start her shopping while Sam is calm interacting with hands-on learning software displayed on the cart's screen. Normally, Susan buys different healthy fruits and vegetables such as bananas or apples. Once Susan has inserted a banana in the shopping cart, different articles appear on the screen in front of Sam. The screen shows a picture of a banana but also pictures of other articles such as an apple or a pizza (see figure 2 (left)). Now, Sam has to select the correct picture by simply touching it. As a feedback, a smiley is displayed which represents the correct or wrong response to the selection (see figure 3). This contextual learning game is considered to train a toddler's mapping skills of real and virtual objects. Other possible game types are illustrated in figure 2 (center, right). The example illustrates the potential of contextual learning games. They can hands-on train toddlers' skills and knowledge. Another example is *Memory*. Some of the articles in the shopping cart are displayed as cards, which have been turned over. The task of Sam is to find the correct pairs. In this application, Sam's memory skills can be trained. Overall, we think that contextual learning games on instrumented shopping carts support a promising novel platform for human computer interaction. However, several design challenges are raised.

Design Challenges

Several design challenges come up when developing user interfaces for toddlers. The challenges can be

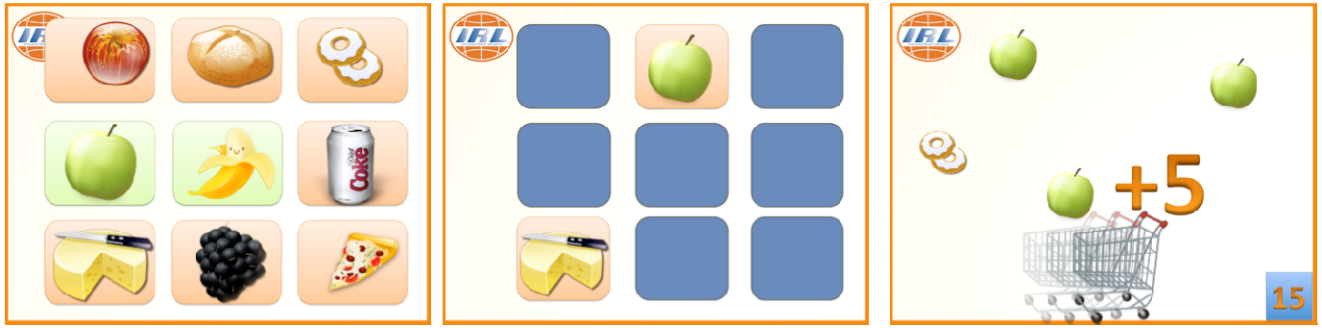

Figure 2: Example for contextual learning games (from left to right): "What have we brought so far?" (game concept described in the scenario), Memory with "real world items" and "Get the apple" (the toddlers have to collect as many apples as they can that fall down by moving the small shopping card from left to right).

summarized in the following two categories: toddlers' skills and design process issues.

Toddlers' Skills

The first design challenge is the consideration of the toddlers' skills. As people develop from infants to adults, their abilities increase over time [7][12][13]. Abilities can be divided into:

- Cognitive Skills and
- Physical Skills.

Toddlers' cognitive skills are often not as trained as those of older children. Hence, the interface designers must consider several aspects. For example, the required memory load must be kept as low as possible, e.g. by reducing the number of displayed items. Another aspect concerns the perception and concentrativeness of toddlers. Toddlers' response time is lower, than that of older children, which causes longer interaction times. Thus, interface designers have to provide clear and easy to recognize information to support the toddlers' perception and reduce response times. To increase the children's concentrativeness, the user interface has to be easy to use but at the same time absorbing to keep the concentration of the children. A further cognitive aspect affects the lack of toddlers' literacy skills. Accordingly, user interfaces have to provide widgets with icons instead of text. These icons must indispensably follow toddlers' knowledge to assure the correct interpretation of an icon's meaning. For example, different expressions of a smiley can mean the positive or negative responds to an interaction. The question is whether children at age of two or three can understand this feedback.

Apart from toddlers' cognitive skills, physical skills such as motor skills must also be considered. A lack of advanced motor skills can cause problems when interacting with widget displayed on the screen. For example, trained motor skills are required for the correct positioning of a widgets. Thus, interface designer should provide interactions which require less motor skills such as simple clicks on large widgets and easy to perform drag and drop operations.

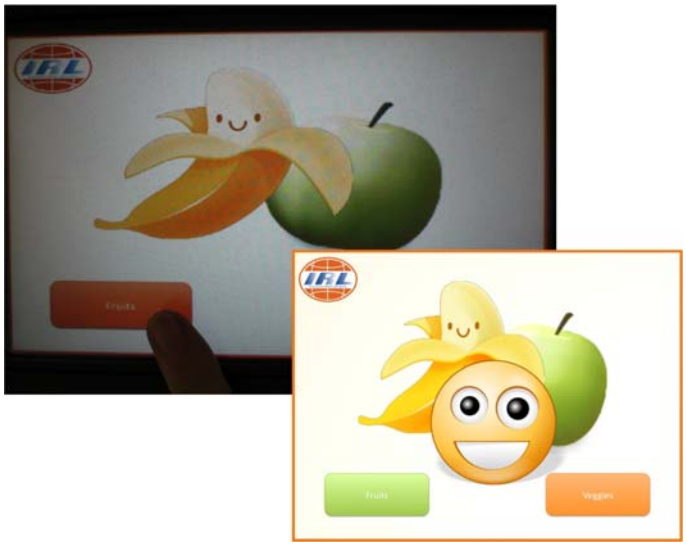

Figure 3: A kid interacting with the shopping card: “What’s in? Fruits or veggies?”

Design Process

The user-centered development requires test with real users in the different phases of the development process. Hence, children at the age of two or three years are required to investigate user interfaces. We consider problems in recruiting users for several studies. Whenever the target group is very restricted to different characteristics, such as age it is time-consuming to find and recruit enough users to run evaluations. When conducting user studies with toddlers, even more challenges emerge. In our described scenario, toddlers are tired and grouchy when interacting with the applications. Thus, we require toddlers with a defined behavior, which can often not be artificially aroused. Another issue concerns the execution of applied tasks. Toddlers might not be disposed to follow the evaluator's instructions. Hence, controlled studies without any biases can often not be performed. Apart from that, several evaluation techniques become unfeasible if user studies are conducted with toddlers. Subjective methods such as interviews or questionnaires cannot be used anymore. Another example is the *thinking aloud method* which is not feasible because of the toddlers' faculty of speech. Accordingly, toddlers' skills exclude any approved usability evaluation technique. Interface designers can only conduct observations to investigate toddlers' behavior. The lack of subjective methods poses a great challenge for the interpretation of the objective data.

Contextual Learning Games

The described design challenges must be considered when developing user interfaces for toddlers. In this section, we describe the consideration of the aspects when developing a contextual learning game for toddlers. Our shopping cart provides a touch-sensitive display, which enables direct and easy to use interactions such as the selection of items via touching. Moreover, the screen can be used for drag and drop operations. These interactions meet aspects of the toddlers' motor skills. To support their cognitive skills, we reduce the number of displayed items on the screen and use icons to meet toddlers' lack of literacy skills. On the screen, we display large and easy to interpret widgets, which also support the toddlers' cognitive and physical skills. Apart from that, another important aspect of our concept is the combination of the real and virtual world, which is considered to support toddlers' mapping skills and meet the toddlers' knowledge. Toddlers can map real objects of a supermarket such as

a banana in the shopping cart to a widget displayed on the screen. Accordingly, the combination of the two worlds can help toddlers to interpret and relate real aspect to virtual information. Our approach of an instrumented shopping cart will be evaluated with real users in field studies. Thus, we do not have the problem of the recruiting of users because we conduct the studies in real settings of a supermarket. Moreover, the conduction of studies can be performed under a real contextual setting. We can find toddlers and parents in the required mood as described in our scenario. Thus, the results of our studies can provide valuable data to enable correct interpretation of lab studies.

Setup and Implementation

As an application platform for the toddler games introduced in this paper, we decided to use the instrumented shopping cart developed at the Innovative Retail Lab (IRL). The handle of this shopping cart is fitted with a 7-inch touch screen, a finger print scanner and a single button (see figure 4). In a current shopping cart application, the touch screen is used to display the customer's shopping list and information about products placed in the cart. In order to be able to display personal information, such as a shopping list, customers have to enroll their fingers using the finger print sensor. The integrated button is designed to be used for switching back to an application selection view (“home” button). Additionally, the cart is instrumented with two RFID antennas and corresponding tag readers. One of the readers recognizes passive RFID tags placed below the flooring of an instrumented supermarket (which is simulated in the ABC (blinded for review)) and thus enables an indoor tracking of the shopping cart. Based on this location detection, the customer can

Figure 4: The intelligent shopping card: The handle of this shopping cart is fitted with a 7 inch touch screen, a finger print scanner and a single button (photograph taken from the back, left). An RFID antenna is placed under the product basket to recognize which products are placed inside the cart (from below, right).

be navigated to products he or she is searching for. The second RFID antenna is placed under the product basket and is used to recognize which products are placed inside the cart. For this purpose, the products have to be fitted with passive RFID tags. Once a product is recognized in the basket, information about it is displayed on the integrated touch screen and the corresponding shopping list entry is checked. Currently, our system utilizes the Feig high frequency readers and tags[1]. All system components are connected to a computer, which is also integrated in the shopping cart. In the current prototype of the instrumented shopping cart, the touch screen is integrated in the handle in such a way that the customer pushing the cart can comfortably see the displayed content. For children sitting in the shopping cart, this screen orientation is of

[1] http://www.feig.de/index.php?lang=en

course unsuitable. Hence, in order to enable children interaction with the screen, the handle has to be redesigned so that its orientation can easily be changed or an extra screen is attached. In this way, the grownup customers can use the assistance applications that help them manage their shopping and if they need distraction for their toddlers, parents can rotate the display towards their children sitting in the trolley and start an appropriate game.

A second challenge that has to be addressed is the childproofness of the application. The toddlers should not be able to manipulate the application data of their parents. Accidentally changed or deleted customer settings would be an annoyance. Therefore, the application should offer different grades of permission rights. Obviously, it would be appropriate to use the already integrated finger print sensor in order to manage access rights to different applications. Once a toddler game is started, the customer will have to enroll his or her finger to get the right to switch to another application. In this way, an accidental misuse of the application can be avoided.

Conclusion and Future Work

We have presented a concept for a contextual learning gaming designed for toddlers using an interactive display attached to a shopping cart handle bar. We present design challenges and different sorts of games that can be played by the toddlers to let them participate in the shopping process to a certain degree without annoying their parents. In addition, we present a hardware solution for the realizing of such interactive shopping cart games and show design challenges for the hardware setup as well.

Obviously, the next step is running some evaluations with parents and toddlers. We already had some young parents involved in the design process. They gave us very promising feedback and stated that they would love to have such an application. They really liked the idea of reflecting real world items in the games instead of just showing “standard comics”. In their opinion this would be an interesting balance between distracting their kids and involving them in the shopping process by creating a meaningful in-context learning experience.

References

[1] C. Atkin. Effects of television advertising on children. Children and the faces of television: Teaching, violence, selling, pages 287–305, 1980.

[2] J. Brand and B. Greenberg. Commercials in the Classroom: The Impact of Channel One Advertising. Journal of Advertising Research, 34(1), 1994.

[3] A. Butz, M. Groß, and A. Krüger. Tuister: a tangible ui for hierarchical structures. In Proceedings of the 9th international conference on Intelligent user interfaces, pages 223–225. ACM New York, NY, USA, 2004.

[4] K. Cheverst, D. Fitton, and A. Dix. Exploring the evolution of office door displays. Public and Situated Displays: Social and Interactional aspects of shared display technologies, page 141, 2003.

[5] C. Ebster, U. Wagner, and D. Neumüller. Children’s Influence on In-store Purchases. Journal of Retailing and Consumer Services, 26(2):145–154, 3009.

[6] M. Helander, T. Landauer, and P. Prabhu. Handbook of human-computer interaction. Elsevier Science Inc. New York, NY, USA, 1997.

[7] R. Kail. Developmental change in speed of processing during childhood and adolescence. Psychol Bull, 109(3): 490–501, 1991.

[8] M. Kranz, D. Schmidt, P. Holleis, and A. Schmidt. A Display Cube as a Tangible User Interface. In In Adjunct Proceedings of the Seventh Internation Conference on Ubiquitous Computing, 2005.

[9] K. Leichtenstern, M. Kranz, P. Holleis, E. Lösch, and E. Andre. A tangible user interface as interaction and presentation device to a social learning software. In Fourth International Conference on Networked Sensing Systems (INSS '07), 2007.

[10] M. Lindstrom. Branding is no longer child's play! Journal of Consumer Marketing, 21(3):175–182, 2004.

[11] D. Merrill, J. Kalanithi, and P. Maes. Siftables: towards sensor network user interfaces. In Proceedings of the 1st international conference on Tangible and embedded interaction, pages 75–78. ACM New York, NY, USA, 2007.

[12] L. Miller and P. Vernon. Developmental changes in speed of information processing in young children. Developmental Psychology, 33(3):549–554, 1997.

[13] J. Thomas. Acquisition of Motor Skills: Information Processing Differences between Children and Adults. Research Quarterly, 51(1):158–73, 1980.

[14] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, and M. Weiser. The PARCTAB ubiquitous computing experiment. KLUWER INTERNATIONAL SERIES IN ENGINEERING AND COMPUTER SCIENCE, pages 45–97, 1996.

Ubibot – Prototyping Infrastructure for Mobile Context-Aware Computing

Erwin Vedar
Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093 USA
evedar@cs.ucsd.edu

W. Brian Evans
Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093 USA
wbevans@cs.ucsd.edu

William G. Griswold
Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093 USA
wgg@cs.ucsd.edu

UbiComp 2009, Sep 30 – Oct 3, 2009, Orlando, FL, USA

Abstract
Mobile context-aware computing requires lots of infrastructure, making significant term-long student projects difficult to carry out. In this paper we describe Ubibot, a flexible publish-subscribe framework implemented on top of instant messaging. Using instant messaging as a network substrate provides many system- and application-layer network functionalities essentially for free. Ubibot's publish-subscribe mechanism includes operators for placing and delegating computations, making it relatively easy to add services to a running system and alter its structure.

Keywords
Mobile computing, context-aware computing, infrastructure, publish-subscribe.

ACM Classification Keywords
K.3.2 [Computers and Education]: Computer and Information Science Education--miscellaneous; D.2.11 [Software Engineering]: Software Architectures--domain-specific architectures.

Introduction

The emerging field of mobile context-aware computing (MCAC) is exciting to students and researchers. However, there are many challenges to experimenting with or otherwise rapidly developing mobile context-aware software. In contrast to desktop computing, mobile devices are characterized by small screens, low memory capacity and computational power [6, 13, 16]. Networking also tends to be unreliable and low in bandwidth due to device designs and mobility [1, 6, 12, 16]. Experimenters need to account for these resource constraints, or risk creating software that is unusable or drains the device. The network connections for these devices are low speed and not very robust. Experimenters cannot assume a constant connection, or a static arrangement of clients and servers.

Yet, coping with these issues from the earliest stages of development is counterproductive. These issues are further frustrated in research and educational settings, where timelines are tight, and not all design issues are equally interesting (or necessary) to explore. Ideally, students could prototype a basic application that validates the basic functionality, and then add support for the above mobility considerations later in development, once the basic features had been worked out.

The problem space of mobile computing is essentially that of distributed systems, in which participants connect and disconnect during operation, possibly at a different access point [4, 12, 13]. Thus the challenge in solving problems for this space is to create systems and applications that can function in a dynamic environment [6]. Connections to wireless networks are naturally mobile, with weak connections [12].

The ultimate goal for addressing these mobility problems is to create a meaningful façade for the user. Users expect their devices to just work as they move around; the connection and disconnection to the network should have as little consequence for the user as possible. If an application is unable to function once partitioned from the network, it should still handle the situation in a graceful way. Cooperating applications need to be able to find each other in the network, regardless of their host device or the device's physical location in the network.

A unique aspect of mobile computing is context awareness. A key challenge here is converting raw sensor information to a form that is usable by the application and user. There can be multiple sensors for a given context, which are distributed, unconventional, and heterogeneous [17, 18]. These sensors communicate in a naturally asynchronous fashion [17] and the data that they produce is strongly time-dependent [7]. Furthermore, the array of available sensors is evolving [14].

Like mobility, the approach to the problems introduced by context-awareness is again abstraction. In this case, it is a matter of taking raw data and abstracting it to a form that can be acted upon programmatically. This involves dealing with multiple data types, and writing programs that can act upon context information.

Several attempts at lowering these barriers have already been made – mostly focusing on context awareness – ranging from toolkits to complete programming suites. Here we discuss just a few as examples. The Context Toolkit hides much of the

specifics of interacting with low-level sensors in order to enable context-aware applications [8, 9, 18]. It is based on the idea of widgets, which abstract away the details of context sensors and the management of context data to a more easily used form. This approach enables developers to create reusable solutions for low-level sensing mechanisms, then separately use those solutions to create applications. In ActiveCampus, Griswold, et al., extended beyond the idea of the toolkit to explore the issues of extensibility and integration among context-aware applications [14]. Du, et al., placed more focus on the development process [10]. The work aimed at creating an entire suite for developers to work, including a development environment. In effect, his work is complementary to ActiveCampus; it addresses the human problem rather than the technology problems. Topiary addresses the human problem as well, targeting application designers rather than software engineers [15]. It attempts to allow experimentation with different interactions during early stage design, and to allow rapid iteration utilizing user feedback.

In complement to these efforts, our goal is to enable learning through experimentation more quickly and easily by supporting *incremental* development.

Ubibot

We hypothesize that many of the problems of mobile computing cited above can be addressed by an extensible, dynamically reconfigurable publish-subscribe architecture.

The well-known publish-subscribe architectural pattern decouples producers and consumers of context information through asynchronous event-based communication, insulating their activities from the low network speed and (lack of) robustness typical of mobile networks [5, 11]. A publish-subscribe architecture naturally enables a system to evolve along with the sensing capabilities of mobile devices by incorporating new types of events for publication.

However, these capabilities alone are not enough to support MCAC. The addition of dynamic reconfiguration allows devices to introduce new computational functions at low effort, move computations close to their data source, or offload them onto the network to, say, compensate for the limited resources of small, battery-powered mobile devices. In aggregate, these features comprise a flexible and easy-to-use experimentation platform for mobile devices.

We decided to test our hypothesis by creating the UbiBot infrastructure in C#.Net. An experimenter who wants to add a new capability or create a new application need only create a service – a semi-autonomous plug-in using the libraries included with UbiBot. The libraries provide the publish-subscribe communication architecture, and the means for clients to perform dynamic reconfiguration. The libraries are designed to be extensible with new data types. Ubibot, however, does not address the costs and challenges of developing user interfaces on mobile devices.

Ubibot's core feature is the *service*. A service is a semi-autonomous computational unit that has the capability to *publish* events of a prescribed type, as well as *subscribe* to events of a type as prescribed by another service. A full-blown *application* consists of one or more services. For example, a mobile phone application may provide services for location, sound

capture, image capture, etc. When an application consists of just a single service, it may be referred to simply as a service itself.

Ubibot is built on top of instant messaging (IM). Instant message provides location-independent naming of services, asynchronous messaging, and routing through firewalls. This makes it relatively easy to set up a basic MCAC system from scratch. All that is needed is the creation of new IM accounts for each of the relevant nodes in the system (participating fixed computers or mobile devices). It is not necessary to set up a centralized server or related software. Nodes can host one or more services. A message processor watching the IM channel parses incoming messages (event objects) to recognize the destination service and route appropriately. Messages between services on the same node are handled with internal routing, avoiding the IM channel.

What makes Ubibot unique is its ability to "program" the network through *hosting* and *delegation*, enabling developers to start with a very basic system, and then gradually evolve it into a more mature form.

Hosting

When a "client" service subscribes to a "server" service, the server may optionally prescribe that the client host an additional service on its device, essentially a plug-in. This plug-in, running locally, can do things for the client that a purely remote service cannot do effectively. For example, it may create and operate a user interface, or it may run a computation locally that would be (more) expensive to run over the network, such as computing the running average of a sensor value that the client is publishing at a high data rate, and re-publishing the average at a low data rate. This mechanism is a bit like software agents, and is inspired by Fulcrum [2, 3].

Delegation

Sometimes a client may be able to provide a service to others, but with high power cost and low reliability, due to the frequent use of the unreliable network. An example is a client publishing its location to all subscribers (See Figure 1). A more efficient solution would be for the client to publish its location once, and to have a non-mobile service store and distribute this information to other services (See Figure 2). However, such a solution could incur substantial development delays during initial rapid prototyping. Ideally, the simple but ineffective location service could be developed initially, and then later the more robust and efficient service could be added in later – at low cost – if desired.

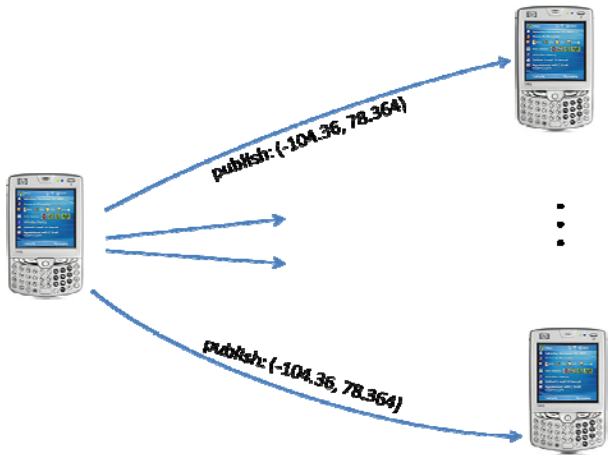

Figure 1. Devices can simply subscribe directly to each others' locations, but puts a lot of network load on publishers.

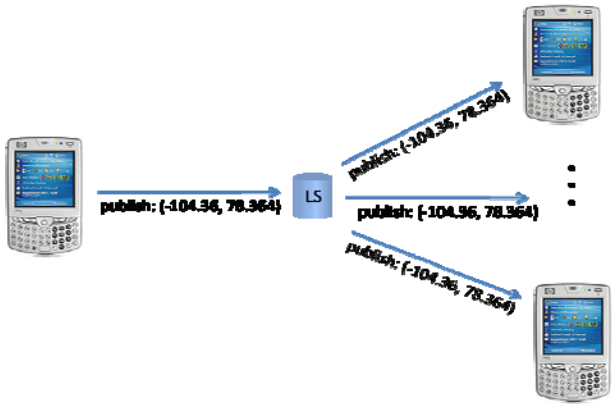

Figure 2. The device on the left has delegated its location subscriptions to a proxying locations service, "LS", shedding network load.

Ubibot's *proxy* operator makes this possible. In the above scenario, the human operator of the naively-developed client would notice that someone had developed and deployed a location distribution service, and then direct the client to delegate location to the new service. This sets in motion the following changes to the system's configuration:

1. The client subscribes to the new location service as a "producer", rather than a "consumer" with the command *proxy location*, along with a list of existing location subscribers, if any, as well as event types that it advertises, e.g.:

   proxy location [{latitude, longitude}, GSM] …

2. The location service then subscribes to the client's location:

   subscribe location [GSM]

3. The client now publishes its location only to the new location service (indirectly as GSM events in this case), rather than to the pre-existing subscribers. The new location service stores this location and republishes the location to the subscribers in the client's stead as *{latitude, longitude}* or *GSM*, just as the client did. Even if the client drops off the network, the location service can continue to provide the last-reported location, albeit with an ever-more-stale timestamp.

4. New location subscriptions received by the client are forwarded to the location service.

Because delegation enables a service to act on another's behalf, some kind of authentication mechanism is necessary to prevent fraudulent delegation. This consideration is beyond the scope of the current paper.

These capabilities of delegation (and hosting for that matter) require enhancements to the traditional publish-subscribe prototype. Just as services *advertise* the event types that they publish, they must also advertise that they can support hosting and delegation. Delegation, however, is not an independent capability, but rather a meta-service. It asserts that if given permission to subscribe to a services type, it can publish that to others. Thus, rather than advertising *location*, for example, it advertises *proxyable: location*, signaling that rather than accepting direct subscriptions to its location, *subscribe location*, it accepts requests to provide others' locations, *proxy location.*

In this delegation example, it is important to note that the client has benefitted from the delegation in several ways:

- it no longer has to compute *{latitude, longitude}*; it can turn off its GPS unit or bypass its GSM → location translation facility, whichever method it was employing;

- it uses the network much less to communication its location, decreasing power consumption;

- if the client drops off of the network briefly, its location remains available to its subscribers.

Case Study

As a preliminary evaluation of the Ubibot concept, two students taking a project class were recruited to develop a location-based reminder service for Windows Mobile phones (HP 6945's). As a twist on the usual reminder service, a reminder could be sent to a buddy, not just oneself. Reminders could be limited to being delivered within a given time range, and could be reset for later delivery if delivered at an inconvenient time.

The students divided their application into server and client components. The client was developed as a plug-in that could be hosted by a hosting service on a phone. The hosting service provided a skeleton GUI that permitted a "tab" to be plugged in to its display. The pre-existing client framework also consisted of services that published the phone's GPS location and GSM observations.

At a high level, an instance of the application is set-up by the phone's client hosting service, by sending *subscribe reminder* to the location service (this may be initially typed in by the phone's user or read from a simple scripting file). The service sends back a handle to the plug-in, which the client loads. The service also subscribes to the phone's location. (Alternatively, the loaded plug-in could have subscribed to the phone's location, allowing it to filter unneeded location events, when the phone isn't moving significantly.)

The students were advised to take an incremental development approach, starting with the most basic self-reminder application, later adding time windows for reminders, and then the ability to remind buddies. For this latter "fancy" feature, its core was simply to subscribe a chosen buddy to a specific reminder, rather than one self. The last incremental enhancement to the application was to delegate the location, as in the previous example. With a single statement of code, the students delegated the GPS location calculation to a GSM → {latitude,longitude} Ubibot service that had been developed earlier on top of Google's hidden API for its "My Location" function of Google Mobile Maps.

The server component consists of five classes, comprising 469 non-blank non-comment lines of code. The client component consists of four classes, comprising 551 lines of code (excluding GUI code generated by Visual Studio), for a total of 1020 LOC. About half of the client code is GUI code, and the students reported that most of their time was spent on getting the GUI to work properly, which required the use of C# delegates, a concept unfamiliar to them as Java programmers. Still, the students had little trouble completing the project in the 10-week class term, and most of their effort was concentrated into a few of weeks scattered across the quarter.

The ability to employ incremental development, especially in adding the “fancy” features of the application (reminders to buddies and delegated location calculation), allowed the professor (the third author) to define concrete milestones throughout the quarter that could be verified with a running demonstration, avoiding ugly surprises and saving all but the last feature from the (unfulfilled) prospect of failure.

Conclusion

Publish-subscribe addresses many of the challenges of MCAC, by providing an abstraction of sensors and communication that minimizes dependencies among nodes. Adding the features of *hosting* and *delegation* provides an incremental development model for publish-subscribe that supports prototyping, enabling students and researchers to initially focus on application features, and later address issues of performance and robustness.

Acknowledgements

This project is supported in part by UC MICRO 07-067 in cooperation with Microsoft Research, as well as a gift from Microsoft Research ER&P and the donation of phones by Hewlett Packard. We thank John Egan and Mark Gahagan for their efforts in developing the location-based reminder program.

References

1. Emmanuelle Anceaume, Ajoy K. Datta, Maria Gradinariu, and Gwendal Simon. Publish/Subscribe Scheme for Mobile Networks. In Proceedings of the second ACM international workshop on Principles of mobile computing (POMC '02), pages 74-81, October 2002.

2. R. T. Boyer and W. G. Griswold, "Fulcrum - An Open-Implementation Approach to Internet-Scale Context-Aware Publish / Subscribe", HICSS'05: Software Technology Track, Proceedings of the 38th Annual Hawaii International Conference on System Sciences, January 2005.

3. R. T. Boyer,"Open-Implementation Approach to Internet-Scale Context-Awareness", Ph.D. Dissertation, June 2005.

4. Mauro Caporuscio, Antonio Carzaniga, and Alexander L. Wolf. Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications. IEEE Transactions on Software Engineering, 29(12):1059-1071, December 2003.

5.Carzaniga. A., Rosenblum, D. S., and Wolf, A. L., Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service, 19th ACM Symposium on Principles of Distributed Computing, pp 219-227, 2000.

6. Oleg Davidyuk, Jukka Riekki, Ville-Mikko Rautio, and Junzhao Sun. Context-Aware Middleware for Mobile Multimedia Applications. In Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia (MUM 2004), pages 213-220, October 2004.

7. Thanos Demiris. Context Revisited: A brief survey of research in context aware multimedia systems. In Proceedings of the 3rd international conference on Mobile multimedia communications (Mobimedia ‘07), 2007.

8. Anind K. Dey and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Aware Applications. In the Workshop on Software Engineering for Wearable and Pervasive Computing , Limerick, Ireland, June 6, 2000.

9. Anind K. Dey, Daniel Salber and Gregory D. Abowd. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Special issue on context-aware computing, *Human-Computer Interaction (HCI) Journal*, Volume 16 (2-4), 2001, pp. 97-166.

10. Weichang Du and Lei Wang. Context-Aware Application Programming for Mobile Devices. In Proceedings of the 2008 C3S2E conference (C3S2E '08), pages 215-227, 2008.

11. Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, Anne-Marie Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2): pp. 114-131.

12. Umar Farooq, Shikharesh Majumdar, and Eric W. Parsons. Engineering Mobile Wireless Publish/Subscribe Systems for High Performance. In Proceedings of the The IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS '04), pages 295-305, 2004.

13. Abdulbaset Gaddah and Thomas Kunz. A Proactive Mobility Extension for Pub/Sub Systems. In Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications (MOBILWARE '08), February 2008.

14. William G. Griswold, Robert Boyer, Steven W. Brown, and Tan Minh Truong. A Component Architecture for an Extensible, Highly Integradted Context-Aware Computing Infrastructure. In Proceedings of the 25th International Conference on Software Engineering, pages 363-372, May 2003.

15. Yang Li, Jason I. Hong, and James A. Landay. Topiary: A Tool for Prototyping Location-Enhanced Applications. In Proceedings of the 17th annual ACM symposium on User interface software and technology (UIST '04), pages 217-226, October 2004.

16. Gero Muhl, Andreas Ulbrich, Klaus Hemann, and Torben Weis. Disseminating Information to Mobile Clients Using Publish-Subscribe. Internet Computing, 8(3):46-53, May-June 2004.

17. Ricardo Couto A. da Rocha, Markus Endler. Evolutionary and Efficient Context Management in Heterogeneous Environments. In Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing (MPAC 2005), pages 1-7, November-December 2005.

18. Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit (CHI '99), pages 434-441, 1999.

Teaching Smart Environments and Co-operative Ensembles

Sebastian Bader,
Thomas Kirste &
Christoph Burghardt
MMIS,
Department of Computer Science
University of Rostock
18059 Rostock
Germany
FIRST.LASTNAME@uni-rostock.de

UbiComp 2009, Sep 30 – Oct 3, 2009, Orlando, FL, USA

Abstract
Teaching in the area of ubiquitous computing provides an interesting and challenging task. The teacher has to select an appropriate amount and level of material to be presented during the course and the practical parts have to be prepared. During practical sessions accompanying the courses, many implementation and hardware related problems have to be solved. Here, we present a course taught at our university on a master-program level, i.e., all students already had some background on computer science, but not necessarily on the subjects presented during the course. We describe the general idea, the topics covered during the theory lectures and the setup chosen for practical sessions.

Keywords
Teaching Ubiquitous Computing, Smart Environments, Co-operative Ensembles

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI): Miscellaneous, I.2.11. Distributed Artificial Intelligence: Multi-Agent Systems, K.3.2. Computer and Information Science Education: Curriculum

Introduction

During the winter term 2008 / 2009, we taught a course called *"Smart Environments and Co-operative Ensembles"* (SE & CE). The course has been given on a masters-program level (German "Diplom") and one term in Germany lasts 14 weeks in total. All students had a background in computer science, but a background on ubiquitous computing has not been assumed. This paper provides a short description of this course: its theoretical content and the practical work accompanying it. It should be seen as report on work in progress because it has been the first time we taught this course. Even though it was the first time we taught it, we regard the course as quite successful, resulting in a number of interesting technology demonstrations developed by students during the practical part.

This paper is organized as follows. First we describe the general idea of the course including some background information on the environment used to teach in. Afterwards, we outline the theoretical content covered during the lectures. Then we discuss the practical part, i.e., the project work accompanying the course. Finally, we summarize the lessons learnt while teaching SE & CE and draw some overall conclusions.

Idea of the Course

In our research, we focus on several aspects of smart environments and in particular on *activity recognition* and *device co-operation* within ad-hoc ensembles. To teach some of the basic principles, we designed the course described here.

We intended to cover two topics, namely multi-agent systems and smart environments in general. And we wanted to balance lectures on theory and practical sessions. Those dual aspects are described in some more detail below. Afterwards and before diving into the details of the theoretical content of the course, we briefly outline our smart environment and its capabilities.

Multi-Agent Systems and Smart Environments

The communication between and the interaction of different agents situated in a shared environment plays a central role in the area of device co-operation in smart environment and thus also for the field of ubiquitous computing in general.

The theoretical content of the course was split into two parts, one focusing on multi-agent systems and the second focusing on some aspects of smart environments in general. As described below each part has been accompanied by practical assignments as well as student presentations.

During the first part, we taught the basic principles of intelligent agents and of multi-agent systems, already with some focus on implementation. Therefore, we did not teach the full theory behind the scenes, but provided only the basic ideas and some important consequences.

The second part has been on smart environments. Again not detailing the theoretical foundations, but concentrating on the underlying ideas. In particular, we taught some basic principles on planning in general, the basic methods underlying semantic technologies, and the basics of activity recognition using probabilistic models like hidden Markov models [14]. The details of both parts are discussed below.

Theory and Practice
As mentioned above, we interleaved theoretical lectures and practical work. During the first part on multi-agent systems, the students were taught the basics of MASs during usual lectures. As assignment, they had to study different MAS implementations in small teams, one system per student team. During a presentation session, each team had to present their system and a small demonstration showing its usage. For the presentation the students have been asked to describe their system with respect to the key ideas presented during the lectures.

The second part of the course has been on smart environments in a more general setting. Here, the students had to implement some new feature for our lab with the goal to make it smarter or to add a new functionality to it. We did not preselect any topics, but let the student come up with their own ideas. (Both parts are further detailed below.)

In both course sections, there was an equal split between theory and practice: there has been a lecture (90 minutes) every week and the students has been asked to spend the same amount on time for their implementations.

Our Smart Environment
Since our smart environment played a central role during the course, we provide some details of it here. Our lab is a smart meeting room, equipped with numerous sensors and actuators. Furthermore, we have a middleware of our own design, providing an easy to use access to all devices and their functions.

The lab is equipped with a “fish-eye”-cameras providing an overview of the whole lab and two pan-tilt zoom cameras, which can focus on almost all details within the lab. It also features a Ubisense [5] system to track the position of the users. The students had also access to accelerometers [4], Wiimotes [2], and a brain-computer interface [3].

All screens, sun-shades, lamps, some electric plugs, and the projectors can be controlled remotely. I.e., the environment can easily be customized to the current needs. Every wall, screen and sun-shade can be used as a projection surface and the VGA outputs of all computers can be redirected to any of the projectors. One of the projectors is furthermore equipped with a remote-controlled mirror, allowing to project a moveable image onto the floor, the tables and all walls (basically, it is a kind of everywhere display).

Our middle-ware provides easy access to the infrastructure of the room. All functions can be controlled using simple command strings sent to certain sockets. E.g., sending the string `"Lamp 1 toggle on"` to this socket would switch on the first lamp in the lab. Similarly, the user can control all other devices like sun-shades, screens and projectors.

Theoretical Content
As mentioned above, the focus of the course has been twofold. On the one hand, we tried to explain the theory behind smart environments, on the other hand, we wanted to provide a hands-on experience for our students. During the (theory)-lectures, we covered the following topics:

Intelligent Agents
In the first lectures, we introduced the notion of an intelligent agent. To do so, we basically followed “Artificial Intelligence: A Modern Approach” [15]. After defining the general notion of an agent, we focused on properties of agents and their environment. The main goal of this first part has been to teach the basic principles underlying intelligent agents and general approaches to model them.

Multi-agent systems
After teaching intelligent agents, we moved towards multi-agent systems (MAS). For this, we used “An Introduction to Multi-Agent Systems” [17]. In this part we focused on two aspects of MASs, namely how communication between agents can be organized and how agreements can be reached.

Communication between agents
The communication between different agents can of course be done using different methodologies. In this course, we showed how it can be realized using *Contract Net* [1] and *Blackboard Systems* [9]. We furthermore covered the *Speech Act Theory* [16] and agent communication languages like *KQML* and *KIF* [8].

Reaching agreements
To show how agreements can be reached we discussed different forms of *auctions* and talked about *task-oriented domains* as a form of “mechanical” negotiation.

After covering the theory of multi-agent systems, we asked the students to implement a small demo scenario using a given MAS and, if possible, to integrate it with our lab environment. This first project work is described in more details below.

Modal logics
Finally, we introduced modal logic to show how reasoning can be done within the abstract agent architectures described previously. Again, this has been done on a very practical level, not providing the theorems behind the scenes, but by rather showing how to use the logic.

Planning
To synthesize a strategy to support the user, the environment needs to plan its next actions. Therefore, we introduced the situation calculus, STRIPS and PDDL, and discussed typical introductory planning problems [15].

Semantic technologies
As an outlook beyond the implementation of our smart environment, we also gave an overview on semantic technologies. I.e., the core ideas, the problems that can be approached using those technologies, and the basic principles behind the scenes.

Activity recognition based on probabilistic models
Finally, we taught the basic principles underlying our approach for high-level activity recognition, namely probabilistic models and in particular so called *Hidden Markov Models* [14]. Due to a tight integration of this activity recognition infrastructure with our environment, it is easy to design such a model for recognizing activities based on the location of different persons. Environment reactions can then be triggered be the recognition of specific activities.

Implementation Work

The practical sessions accompanying the course have been split into two parts. During the first part, the stu-

dents had to study multi-agent systems and during the second they had to implement their own technology demonstration within our lab. Both parts are described in more detail below.

Project Work: Multi-Agent Systems
After discussing multi-agent systems in theory, the students have been asked to apply them. For this the students teamed up in small groups with 2 to 3 persons each. To every team a MAS implementation has been assigned. The students had to study these system and to implement a small demonstration showing its capabilities. The following MASs have been studied: The Open Agent Architecture [7], Jade [6], Jadex [13], MASON [11] and Ascape [12].

The students have been asked to present their system and if possible a demonstration of it. These presentations have been scheduled during a single seminar meeting in the middle of the semester.

One team of students managed to connect their multi-agent system and our lab, resulting in the demo “Simon Says” as detailed below. The other teams simply showed how to use the assigned system and presented simple examples. By studying one system in detail, all students were able gather experience with the concrete “look and feel” of multi-agent systems in general and their usage.

SIMON SAYS
Based on the Jade-System [6], a team of three students developed a small multi-agent scenario for controlling our lab’s lighting infrastructure. All lamps in the room have been represented by an agent. Those agents have been implemented as simple reflex agents listening for certain commands, namely the commands to switch the lamps on and off, or to dim them to a certain degree. Another agent, “Simon”, has been implemented to send commands to the others using the internal communication system of the Jade-System. Those commands to control the lamps have been sent in more or less random order, resulting in an interesting discotheque-like feeling within the lab.

Even though not that useful in practice, the demonstration showed that it is fairly easy to connect a high-level system with low-level actuators in our lab. And maybe more important for the course, it motivated the other teams to come up with a nice demo during the second practical part.

Project Work: Smart Environments
During the second half of the semester, the students had to develop a small application running in our lab. As mentioned above, we did not specify the topics, but let the students come up with their own ideas. At this point the students knew how intelligent agent systems work in practice and how to use our infrastructure. We furthermore offered to provide any further hardware imaginable (within reasonable budget limits …). The resulting demos are detailed below.

WIIMOTE ENVIRONMENTAL CONTROL
The first team designed a room controller based on a standard Wiimote [2]. Using this controller, the user is able to simply point at a device and control it. Since our room is equipped with eight screens that can be moved up and down and six lamps that can be switched on and off, a normal remote control for it would need at least (8 + 6) x 2 = 28 buttons to control those devices. By simply pointing at the device, we would need only

two buttons. For some practical reasons, the student used 2 buttons for the screens and two different buttons for the lamps, but they could show that the controller does indeed work as expected.

This small demonstration could indeed be further refined into a universal remote control for instrumented environments based on the Wiimote controller. Such a simple-to-use remote control would provide a simple and coherent access to complex environments, which is also cheap to develop.

Playing Tennis in Instrumented Environments

Based on the position tracking in our lab, the second group developed an interactive tennis game. By moving from one corner to the other, the students could control their tennis racket. The movable projector has been used to project the ball onto the floor. The other projectors have been used to create a stadium-like atmosphere by showing an audience, the referee and the scoreboard. Furthermore, the students controlled a pan-tilt-zoom camera to provide a close-up of the currently active player.

A Brain-Interface for Environmental Control

The third team of students bought a simple brain-computer interface, namely the *Neural Impulse Actuator* [3]. This interface, measures some electric potential on the forehead. Those potential include electro-myograms (potentials arising from muscle control), electro-encephalogram (signals from the nerves in the brain) and electro-oculogram (signals from eye movement). Using the controller it is easy to recognize the following actions: eye movement in general, heavy muscle movement on the forehead or moving the jaw, light muscle movement on the forehead, heavy thinking, and relaxing or closing the eyes.

Although these signal are very noisy and often highly correlated, they can be successfully used to trigger simple actions in the room. The students implemented a simple graphical interface allowing the user to navigate through the room's devices, to navigate through the actions performable by the selected device and to finally trigger the selected action. Such a hands-free controller can for instance be useful for impaired people in a wheelchair or lying in their beds, where it might simplify deliberate environment control.

Whiteboard Interaction using Pen, Camera and Projector

As almost all meeting rooms, we have a usual whiteboard in our lab. But this whiteboard is also covered by one of our pan-tilt-zoom cameras and a projector. Therefore, we can pick up what is being drawn onto it by video, and we are able augment the “real” drawings with computer generated projections. Based on this, two students implemented an enhanced whiteboard interaction. For the demonstration purpose, two different settings have been implemented.

The user can draw lines onto the board and afterwards place a number of virtual balls onto it. Those (projected) balls would simply follow the laws of physics and start to roll down along of the (manually drawn) lines. This type of interaction would be a nice feature while teaching physics in school, because it enables the teacher or students, to interact on a very natural base with a physical simulation. Of course, systems with such capabilities have been developed before (e.g., [10]), but they are usually based on a special board,

able to recognize the drawings of the human and not on capturing the scene using a normal camera.

The second mode enables the human to draw a maze onto the whiteboard. Afterwards a number of (projected) lemmings, as known from the computer game[1], enter the scene and again by following the laws of physics and those of lemmings, they move through the maze. Again, this demo is not a computer game which could be sold, but shows new possibilities for interaction in computer games.

AGENT BASED ENVIRONMENTAL CONTROL

Another group controlled the room's lights by means of a multi-agent system. Some virtual agents placed in a simulation of the lab tried to follow the user, i.e., they tried to move in the room to those positions related to the user's position in the real world. While moving in the virtual world, they could switch the lights on and off, which would switch the real lights too. By trying to follow the user and by switching on the closest light, the agents implemented some kind of “follow-me” behavior in the room. I.e., the closest light to the user has always been switched on. Even though this kind of functionality could have been achieved by simpler means, it shows again that high-level principles can be applied to achieve a given goal.

Lessons Learnt

In the introduction we claimed that the course can be regarded as a successful one. This claim can be justified by the following observation: All students have been willingly following the course and the overall motivation has been very high throughout the whole course. But the students have not only been motivated, they have also learnt the basic principles of multi-agent systems and smart environments.

In particular by mixing theory and practice, the motivation has been kept on a high level. And after implementing their demos, the students knew why they had to learn the topics covered during the lectures. Nonetheless, there are a few points, which can and will be improved for the following teaching period. Below we provide a list of things that have been found to be good while teaching the course and a list of things that can be improved.

Good Things

The mixture of theory and practice has probably been the best decision we did while designing the course. We wanted a well balanced course in which the students learn the basic principles and on the other hand get the possibilities to apply them in real-life applications.

Splitting the course into two halves helped also to focus more on the current part. During the first, more theoretical part, the students were taught the fundamentals of agents and multi-agent systems. Afterwards they were required to apply their knowledge while preparing small demonstrations and presentations. These midterm presentations did also help to maintain a high level of attention, because the students had to give a presentation outside the usual examination periods at the end of the semester.

The implementations developed during the course have only been possible due to our simple to use middleware. The salient point here is that our middleware strives to give a simple and streamlined access to

[1] http://en.wikipedia.org/wiki/Lemmings_(video_game)

things that should be simple when *experimenting* with sensors and actuators in smart environments (in contrast to middleware facilities required for *delivering* a smart environment to the end user): A simple, socket-based command environment, using an uncluttered, human-readable syntax, requiring no fancy graphical user interface, is highly instrumental for the quick experimentation with the available devices. Although our middle-ware is part of our ongoing research projects, and still in a state of flux, it has been stable enough for the requirements of the course.

The possibility to develop an own application during the second part of the course had probably a positive impact on the student's motivation also. The only requirement we imposed was that the application must do something with the available infrastructure. Therefore, the student had the free choice while selecting the sensors and actuators, and while designing their application scenario. This resulted, as described above, in a number of completely different technology demonstrations and in a lot of fun during their presentations.

Things to be Improved

Due to the fact that we did not specify the type of application to be developed during the second practical session, we found a substantial mismatch between course content and realized applications. Only one team tried to apply a multi-agent system while controlling the room. Therefore, it could be better to somewhat focus the area from which applications might be chosen. As mentioned above, the free choice had a positive impact on the students' motivation, but we think it is possible to acquire more focus without sacrificing too much motivation.

We believe that focusing the overall scope could improve the course. Some topics covered during the lectures could not be applied while working on the implementations. E.g., the background information on modal logics and on planning has not been applied by any of the teams. Therefore, one could either require that it should be used while modeling the agents or those topics could be left out.

To prepare the same course for the next term, we will first fix some minor problems on our middle-ware and we will probably revise the course content a little. The middle-ware offers a simple interface to execute actions, but does not yet allow to access the current state of the world. Therefore, every agent needs to maintain an internal representation of the world, which is not necessarily consistent with the real world itself. But this can be fixed by adding the appropriate access-methods. As mentioned above some topics have not been covered during the practical parts. This could be enforced by requiring the students to apply the underlying ideas within their demos.

Recommendations

To teach a similar course in a different environment, we propose to equip the environment with a simple to use middle-ware and to design the course such that the theoretical content covered during the lectures is appropriate with respect to the available infrastructure.

In particular, the socket-based and human readable control sequences have simplified the development of the demonstrations a lot. Without our middle-ware, the students would probably have spent most of the time debugging the hardware, which usually results in frustration. More high-level interfaces like web-services or

UPNP do also provide a common interface to all available devices, but the protocol overhead is much bigger. Our socket-based interfaces can not guarantee all security protocols etc., but those are probably not important while teaching a beginners course on ubiquitous computing. Those topics could of course be covered but a simpler interface is probably better since it is easier to use.

Conclusions

This paper contains a description of our course on *"Smart Environments and Co-operative Ensembles"*. In usual lectures, we covered topics like intelligent agents and multi-agent systems, logic and semantic technologies as well as inter-agent communication and negotiation protocols. Small student teams have been confronted with the task to make our instrumented environment a little smarter, which resulted in a number of nice technology demonstration, which are now integrated into our environment.

In this paper we describe both, the theoretical content covered during usual lectures, and the results of the practical parts accompanying the course. Probably due to the mixture of theory and practice and due to the fact that most parts of the theory had to be implemented in student teams, we observed a constantly high motivation among all students.

References

[1] FIPA contract net interaction protocol specification. Technical report, Foundation for Intelligent Physical Agents, 2002.

[2] http://en.wikipedia.org/wiki/wii remote, JUN 2009.

[3] http://www.ocztechnology.com, JUN 2009.

[4] http://www.sparkfun.com, JUN 2009.

[5] http://www.ubisense.de, JUN 2009.

[6] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade - a white paper. Technical report, SEP 2003.

[7] A. Cheyer and D. Martin. The open agent architecture. Journal of Autonomous Agents and Multi-Agent Systems, 4(1):143–148, March 2001. OAA.

[8] T. Finin, Y. Labrou, and J. Mayfield. Software agents, chapter KQML as an agent communication language, pages 291–316. MIT Press, Cambridge, MA, USA, 1997.

[9] B. Hayes-Roth. A blackboard architecture for control. Artificial Intelligence, 26:251–321, 1985.

[10] S. Jeschke, L. Knipping, R. Rojas, and R. Seiler. Intelligent chalk-systems for modern teaching: in math, science engineering areas. In Proceedings of ASEE'06, 2006.

[11] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. MASON: A multiagent simulation environment. SIMULATION, 81(7):517–527, July 2005.

[12] M. Parker. What is ascape and why should you care? Journal of Artificial Societies and Social Simulation, 4, 2001.

[13] A. Pokahr, L. Braubach, and W. Lamersdorf. Multi-Agent Programming, chapter Jadex: A BDI Reasoning Engine. Kluwer Book, 2005.

[14] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286, 1989.

[15] S. Russell and P. Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall, 2 edition, 2003.

[16] J. Searle. Speech Acts. Cambridge University Press, 1969.

M. Wooldridge. Introduction to MultiAgent Systems. John Wiley & Sons, June 2002.

Introducing TU100 ‘My Digital Life’: Ubiquitous computing in a distance learning environment

Mike Richards
Computing Department,
Faculty of Maths, Computing and Technology,
The Open University,
Walton Hall,
Milton Keynes.
MK7 6AA
m.richards@open.ac.uk

John Woodthorpe
Department of Communications and Systems,
Faculty of Maths, Computing and Technology,
The Open University,
Walton Hall,
Milton Keynes.
MK7 6AA
j.woodthorpe@open.ac.uk

UbiComp 2009, Sep 30 – Oct 3, 2009, Orlando, FL, USA

Abstract
In this paper we describe the Open University’s progress towards delivering an introduction to ubiquitous computing within a distance-learning environment. Our work is strongly influenced by the philosophy of learning-through-play and we have taken technologies originally designed for children’s education and adapted them for adult learners, many of whom will have no formal experience of computer science or information technology.

We will introduce two novel technologies; Sense, a drag-and-drop programming language based on Scratch; and the SenseBoard, an inexpensive hardware device that can be connected to the student’s computer, through which they can sense their environment and display outputs.

This paper is not intended as a detailed discussion of individual technologies (they will follow in time), rather it should serve as an introduction to the Open University’s method of teaching and how we hope to continue to recruit new computer scientists and engineers using novel technologies.

Keywords
Ubiquitous computing, distance education, adult learners, programming

ACM Classification Keywords
K.3.2 - Computer and Information Science Education: General Terms. Experimentation, Human Factors, Languages.

Distance learning at the open University

With more than 250,000 active students, the Open University (OU) is Britain's largest university and is currently celebrating its 40th Anniversary. The university offers independently audited qualifications ranging from introductory certificates, through to bachelors and postgraduate degrees and doctorates as well as a range of recognized professional certifications.

All OU undergraduate students study at a distance. Most courses use printed self-study materials that are assessed at regular intervals through the course and at the end. The development of new courses may take a number of years; with materials going through a large number of quality assurance procedures in terms of readability, consistency and accessibility to all students. This course development process is extremely expensive and may cost several million Pounds.

Each student belongs to a 'tutorial group' that is run by a part-time, fully-trained 'associate lecturer'; who in turn is supported by central academic staff at the university's main campus in Milton Keynes. This scalable approach allows the OU to teach very large cohorts of students (some courses may have in excess of 10,000 students during each presentation), whilst ensuring individual students can receive personalized professional support.

Unlike most universities, the OU does not require any previous educational achievements as a condition of entry; instead offering a range of introductory courses teaching a basic grounding in subjects as well as key educational skills which will aid further study. OU student results compare favourably with comparable students from conventional universities.

The OU has always been at the forefront of innovation; not only was it Britain's first distance education institution; but it went on to pioneer the use of educational television and radio programming through a long-standing relationship with the BBC. During the 1990s the OU trialed many of the Internet technologies; such as electronic delivery of materials, computer conferencing and computer marking, which are now commonplace in other universities [1].

Modernising the open university curriculum

The OU has two, extremely popular entry level courses, attracting approximately 4,000 students apiece each year. T175 *'Network Living'* is an introduction to the technologies and role of networks in modern society, whilst Computing offered M150 *'Data, Computing and Information'* is a 'traditional' computing course concentrating on acquisition, processing and use of data.

In mid-2008, a decision was taken to replace the two courses with a single course reflecting the rapid technological and societal developments that had taken place in the last five years. The new course had to fit into an existing degree programme and to deliver certain key learning outcomes required for later study; most notably some experience of computer programming. However, the method of teaching could be novel.

Very early on we recognized that ubiquitous computing would allow us to teach all of our existing learning outcomes but provide a unique opportunity to distinguish our introductory offerings from those of any other university in Britain; where ubiquitous computing was taught at all, it was only ever taught at a higher

level – usually in the final year of study. Ubiquitous computing provided us with an opportunity to widen participation in computer science and engineering. If computers are going to be everywhere and indeed 'everyware', then we should attempt to teach the subject in a manner that would appeal to as large an audience as possible.

Course development would extend over two years with the first presentation scheduled for 2011. Creation of the materials would take approximately one year with six months given over to developmental testing by fellow academics, prospective associate lecturers and a number of existing OU students.

Course Content

From the outset we chose to develop an approachable course that began when we gave it the name *'My Digital Life'*. The course team felt that giving students some sense of ownership would help them engage with the course materials – and so, the obvious place to start the six teaching blocks was with the students themselves and to gradually expand their view until it encompasses the whole World; showing how ubicomp has already started to become a reality [2].

Block 1 - Myself

The course starts with a block concerned with our needs to consume, process and publish information. It introduces students to the concepts of data and information and how these are created, distributed and stored. The concept of personalized information is central to this block; students will create their own online 'home' and fill it with appealing information that they want to share with others. In the process they will learn about the underlying structure of the Internet, the software that interacts with the Internet and the way data is encoded into machine-readable forms.

Block 2 - My stuff

The second block expands our view to encompass the devices we use on a day-to-day basis; from the familiar personal computer to smart phones, games consoles and television smart boxes. At their heart each of these devices is a computer. Uniquely, computers can change their behaviour depending on what software they are running.

The block begins by an exploration of the software and hardware used to create these devices and introduces students to the principles of computer programming. The second part of this block is concerned with how we interact with computers; as well as discussing the discipline of human-computer interaction (HCI), the unit explores how smart devices can find and filter useful information from the overwhelming amount of material in circulation. Students will learn how to find, rank and reference information and continue to build their literacy skills.

Block 3 – My place

Expanding outward again, this block is concerned with an individual user and how they interact with computers in a mobile environment. This block provides an in-depth discussion of ubiquitous computing, using familiar devices such as mobile telephones and GPS (Global Positioning System) devices, and exploring developments including location-based services and Cloud computing. The block closes with a look at the

role of standards and contrasts open and closed source methods of working, and will provide some examples of how different business models and ways of working can co-exist in more than just the area of software development.

Block 4 – My friends

The second half of the course sees a greater emphasis on the social aspects of computer technology. This block discusses how computers allow us to make and maintain friendships no matter how great the distance between individuals. The block is devoted to those relationships we choose to cultivate. It is a discussion of social computing where information is freely shared on a consensual basis. It moves on to discuss the more advanced area of shared spaces that began with multi-user dungeons and is now epitomized by online virtual worlds such as Second Life. The block explores what information people share online and why they wish to share it. It closes with two explorations; the first being how social computing is changing the wider world – in the form of political campaigns fought online; the second being the explosive growth of online video games.

Block 5 – My society

Beyond the people we choose to share information with, there are those with whom we *must* engage online whether we choose to or not. This block is devoted to the growth of the electronic society and how such a society has positive and negative aspects. Case studies are used to show how electronic government can benefit healthcare and to examine the role of the database in enabling such technologies. This block also explores the role of the individual in e-society and how individual rights may conflict with those of the state. This introduces some of the legal aspects of computational technology – ranging from protection of personal data to the importance of copyright. It ends with a provocative examination of whether the freedoms created by modern technology are compatible with the security of individuals and the state. This will show you how to form arguments from conflicting evidence and produce your own, researched, opinion on a controversial topic.

Block 6 – My world

The final block takes the widest possible view of computing technology and demonstrates how it is changing the entire world at an unparalleled pace. We contrast those countries such as Dubai and Singapore that are wholeheartedly embracing the opportunities of computer technology with those parts of the world that face being left even further behind the most developed countries. Our discussion of this 'Digital Divide' is not just concerned with the divide between countries, but that found inside countries – between rich and poor, educated and less educated, old and young. We look at some projects and technologies that aim to close this gap. Finally, the course asks opinions; from technology experts, researchers, science fiction authors and from students, of what the future holds – will computer technology lead to a new utopia or to a dystopia?

Teaching ubiquitous computing

We suspected that ubiquitous computing would prove to be attractive to students – it was new, exciting and slightly quirky. Unlike longer-established areas of computing and technology, many of the ubiquitous technologies being developed are clearly 'rough around

the edges' and there is the still the possibility that a newcomer can make a significant contribution to the field. But we faced two significant challenges;

1. most of our students will come to TU100 without any computer programming experience – let alone familiarity with ubiquitous technologies;
2. because all of our students work at a distance, we *cannot* offer laboratory sessions where they could use the expensive electronics components used in most university ubiquitous computing courses.

Whilst it would be possible to develop a completely theoretical course, perhaps one supported by video material, it would be a tragedy if our students could not get 'hands-on' experience of ubiquitous computing technologies.

Sense

Current first level students studying M150 use JavaScript for their programming exercises. Whilst this is a well-supported, relatively powerful, conventional language; it has proved to be an unsatisfactory choice. Whilst many students learn to program in JavaScript, a significant proportion of users either withdraws from the course or does not progress to more advanced programming courses citing JavaScript's pedantic syntax and relatively poor support for debugging. Student interviews and feedback from associate lecturers suggested that most students could design an algorithm to solve a problem, but lacked the confidence to turn that algorithm into an executable program.

The OU had previous experience through connections to the well-developed RoboFesta movement [3], and in the development of T184 *'Robotics and the Meaning of Life'*, a robotics course for novices [4][5]. Both of these used the LEGO Mindstorms™ kit that could be programmed using the drag-and-drop RCX Code environment. We had found that children and adults found drag-and-drop extremely intuitive to use and were able to build relatively complex programs with rich behaviours.

A decision was made to extend the Scratch [6] language from the Lifelong Kindergarten Group at the MIT media Lab. Scratch is a media-rich programming environment which is especially notable for its clear programming structure; individual program blocks – such as if-else statements, logical operators and variables can only be assembled in meaningful (not necessarily correct) manners, and as such helps remove one of the major frustrations of JavaScript – syntax and logic is implicit rather than explicit.

Whilst Scratch does not offer students an immediately useful language for their employment, it does allow them to learn all of the basic skills needed to succeed in any programming language; it builds confidence and offers plenty of encouragement to study further – unlike most languages, with Scratch *'you don't need to know a lot to do a lot'*. Scratch has proved extremely popular with educators and students alike [7][8].

However, Scratch was not ideal for our purposes; it had been clearly designed for children and might feel patronizing towards adult learners, (although our initial tests with adult volunteers revealed a surprising number of them enjoyed the environment's toy-like appearance). Scratch also lacks some of the richer programming concepts – such as lists - needed to satisfy the learning outcomes needed for further study

at the OU. Most seriously, Scratch lived in a sandbox – it could not talk to the outside world – if we were going to use it to teach ubiquitous computing, Scratch was going to need network support.

Fortunately it is possible to modify Scratch to suit our needs; Scratch is programmed in Squeak and the underlying image file has been made publicly available for modification. Over the last few months we have been building our own programming environment, *Sense*; which is more suited to adult learners exploring ubiquitous computing.

Sense retains Scratch's look and feel, immediacy of operation and welcoming approach that encourages experimentation. Wherever possible we have tried to retain compatibility with Scratch by retaining the names and functionality of individual blocks; but we have added a number of new features such as support for protocols including TCP/IP and RSS, rich data structures and often-neglected commenting.

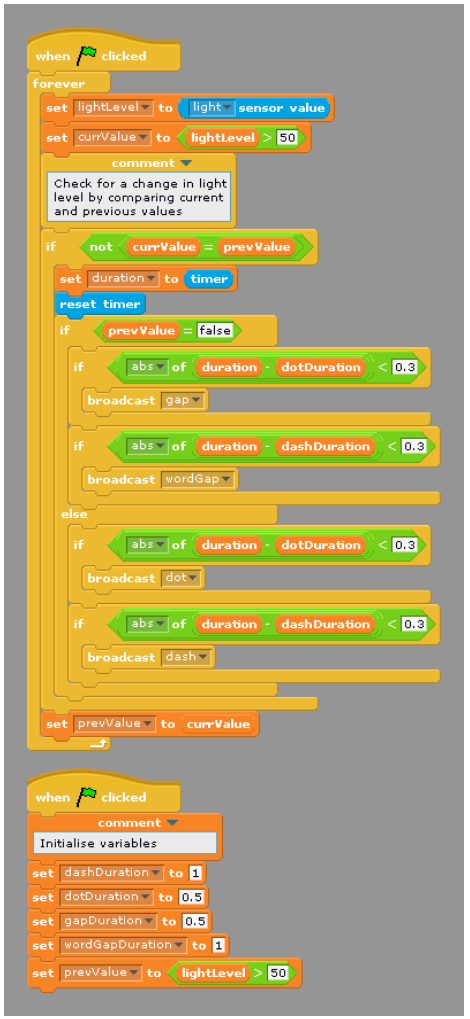

Figure 1. An example of a Sense program showing how program structure is made clear using different shaped blocks and colour.

The SenseBoard

In the past, the OU has provided so-called 'Home Experiment Kits' (HEK) to students but had gradually abandoned their use because of the expense of returning and refurbishing boards when they were no longer needed by students before they could be dispatched to the next student cohort. We decided that this problem could be avoided if the TU100 HEK was made sufficiently cheap that it could be considered 'disposable' so far as the University was concerned. Before committing ourselves to a HEK we assessed a number of 'off the shelf' technologies already used in computer education.

One of the most popular products used in university ubiquitous computing courses is Phidgets [9]; kits of more-or-less plug-and-play electronic components offering almost unlimited potential for experimentation. The high unit cost of Phidgets would have affected the profitability of TU100, but they were finally rejected after feedback from potential students who found Phidgets individual electronic components and breadboards to be extremely intimidating. Further concerns came from members of the course team who were worried about the Phidgets' durability and the possible high rate of returns.

An alternative was the PicoBoard, originally designed by the same team as Scratch, but now marketed by Pico [10]. The PicoBoard is a small printed circuit board that can be connected to a computer through a USB port. The board contains a number of analogue inputs, a light sensor, a microphone, a push button and a slider. Although quite limited in its capabilities, the PicoBoard had a number of advantages over the Phidgets; it was cheap (approximately $50) and robust because of its surface mount construction. However, we felt it was lacking in that it did not offer any outputs such as motors or lights.

With no obvious 'off the shelf' option, the OU commissioned Kre8 Ltd, a company with experience of designing electronic toys to design a low cost sensor board for TU100. The eventual design placed inputs and outputs on a single surface-mount shield that could be plugged into a near-standard Arduino board. The SenseBoard offers a wide range of inputs and outputs for a very low cost (approximately $70 including the Arduino).

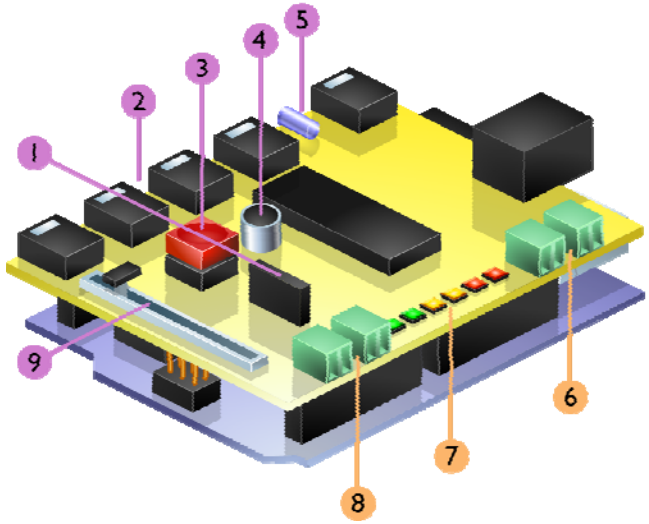

Figure 2. Schematic of the Version 1 SenseBoard (June 2009) mounted on its Arduino host. (Legend below)

Key	
1	IR detector (remote control)
2	Analogue inputs 1 – 4 (temperature, motion, pressure etc.)
3	Push button
4	Microphone
5	Edge IR emitter / detector (barcodes etc.)
6	External motor 1
7	Programmable LEDs
8	External motor 2
9	Slider

Table 1. Key for the SenseBoard diagram (Figure 2).

The SenseBoard will be accompanied by a number of plug-in sensors such as a thermistor and a motion detector, a pair of motors, a battery pack (needed when using the motors), a USB cable and a magnetic board which allows the SenseBoard and all the associated equipment to be carried from room to room or stored without worrying about stray cables or losing items.

An example of a very early project that can be constructed by a novice student in a very short period of time will be to build a simple weather station incorporating a thermistor and a light meter. Initially the student can display the results inside the Sense window on their computer, but after a few more activities they will be able to use Sense to format data into an RSS feed and publish their own results to the Web. They will then be encouraged to look at other students' data and plot their results using technologies such as spreadsheets, online mapping tools or perhaps displaying other results on their SenseBoard using the LEDs or using the motor to drive a pointer.

The aim of the course is not to exhaust the potential of Sense or the SenseBoard, but to demonstrate sufficient of its capabilities that students will be able to develop and share their own projects. Indeed when the student finishes their studies we would be delighted to know that they were continuing to use the board.

Course materials

As well as the SenseBoard, students will receive their course materials as a series of full-colour bound booklets and either one or two DVDs holding all of the course software (including the Sense environment), audio and video interviews, example projects and guided walkthroughs of key tasks. Students will be introduced to their SenseBoards through a number of introductory video 'builds' (much in the style of the popular BBC children's television series *'Blue Peter'*). We will show them how to plug their board in to their computer, test it and then begin programming before moving on to their own projects. All of the bundled materials will be supported by a course Website, containing links to Open University resources – such as technical support, the Library and University's assignment handling system. Students will each have their own personal email box, online file storage area, study calendar, blog and journal. They will converse with their fellows using a range of technologies

including instant messaging and small (20 person) forums, each run by an associate lecturer.

The course is assessed throughout. Six written assignments require students to submit a range of materials ranging from short answers through to researched essays and Sense programs. The course's final assessment takes the form of a short project. Obviously, we cannot expect students to submit their SenseBoards as part of their projects, so students will be graded on their programming code, their documentation and results and we will be asking them to contribute photographic and video evidence to support their conclusions.

Open Source

The developers of TU100 have decided to release much of their work as open source projects.

1. Whilst much of our teaching material cannot be made freely available; some of TU100's content will be placed on the Open University's OpenLearn site [11] for free re-use.
2. The Sense project will be released on a public Web site on an 'as-is' basis.
3. The full specification, layout and component list of the SenseBoard as well as any necessary driver software will be published on a public Web site. Anyone will be able to make their own board without paying royalties, although the board's designers reserve the right to sell their own version of the SenseBoard.

Accessibility

The OU has a long history of ensuring its course materials can be used by anyone who wishes to study its courses. The Sense environment and the SenseBoard raise profound issues for students with visual impairments or limited motor skills. UK legislation requires the OU to make 'reasonable efforts' to allow disabled students to access course materials.

The most basic form of provision requires disabled students to be supported by able-bodied assistants. This is obviously unsatisfactory; not only do some students lack an able-bodied assistant, but also needing to ask for help reduces their independence. The ultimate form of provision is when all materials are usable by all people no matter what type of requirements they have. This is almost always unobtainable, if only because of the great expense in designing appropriate materials. Inevitably some degree of compromise will be required and it is likely TU100 will become more accessible through time as new materials are developed.

We hope to include a certain degree of accessibility support within Sense before TU100 is released. The OU's Institute of Educational Technology (IET) has produced an initial report on Scratch using a range of different assistive technologies and computer settings. IET identified a number of issues that can be summarized as follows:

- People who use a screen reader would not be able to access the application;
- People who do not use a mouse would not be able to operate the application easily;
- People with visual impairments or dyslexia may find some text difficult to read due to low contrast with the background;

- People who require large font size may have difficulty reading the buttons/text;
- People who use a screen magnifier may have difficulty reading buttons text;
- People who use voice recognition would not be able to operate the application easily;
- Hearing impaired people should not have any difficulties, apart from being aware when a sound is played in their program;
- People with a manual impairment may find some of buttons to be small targets;

These issues have been passed to the Sense developers so that a costing for a more accessible version of the application can be presented to the University.

The SenseBoard also presents issues of accessibility. The various connections and input devices obviously present problems for students lacking motor skills. Rather than relying customized components (an approach taken by LEGO with their MindStorms™ kits) which would increase the cost of every SenseBoard to a point where the course was no longer viable; the course team decided to build the board with over-sized components that are easy to grasp and highly robust. Individual items such as power and motor plugs have also been designed in such a way that they can only be connected in the correct manner.

We have also designed an onscreen SenseBoard; the push button and slider of which can be manipulated using the mouse, the SenseBoard's microphone is replaced with that on the computer, and where sensor inputs can be simulated using type-in numeric values.

Visually impaired students do not currently have any accessibility support on the board. However, the course team has entered discussion with the SenseBoard designers to develop a second version of the board. The new board will use speakers and vibration devices to replace the LEDs on the current board. Instead of sequences of light, outputs would be conveyed using musical notes or pulsed patterns of vibrations. We call this device the SenseBoard Touch. This may be a separate board from the original SenseBoard automatically dispatched to any students registered as visually impaired. Alternatively, if the board can be manufactured at a low enough cost, we may issue the SenseBoard Touch to all students so that all students receive a more capable device.

Conclusion

TU100 is a typically ambitious project for the Open University and has required us to develop a number of new technologies to address the particular needs of distance learning students. We have developed a programming environment for adult learners with no prior experience of computer programming, and a complementary piece of hardware that will allow them to begin experimenting with ubiquitous computing.

At a very early stage we realized that a significant number of students may be unable to use some or all of our technologies. The biggest unanswered question remains how we can make ubiquitous computing devices that are truly useful to everyone. We are planning further developments of our hardware and software to not only accommodate, but to actively welcome those people who are normally excluded from education in general, and computing in particular.

In order to facilitate the exchange of ideas between educators we will be releasing much of our work as open-source projects.

ACKNOWLEDGMENTS

We would like to thank everyone on the TU100 course team for all their work so far and for what they're going to do in the future. An especial thank-you to Dr. Chetz Colwell from IET who conducted the initial usability tests on our work. Also to Syntropy Ltd. for their work on the Sense environment. And to Kre8 Ltd., Garry Bulmer and Robert Seaton for the design and layout of the SenseBoard.

References

[1] Carswell, L., Thomas, P.G., Petre, M., Price, B., Richards, M. (1999) Understanding the 'Electronic' Student: Analysis of Functional requirements for Distributed Education. Journal of Asynchronous Learning Networks, vol. 3 issue 1 pp. 7-18 Sloan Consortium.

[2] Bell, G., Dourish, P. (2007) Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. Personal and Ubiquitous Computing, Vol. 11, No. 2. (1 February 2007), pp. 133-143.

[3] RoboFesta Europe (Britain) http://www.robofesta-europe.org/britain/

[4] Price, B., Hirst, A., Johnson, J., Petre, M., Richards, M. (2002) Using robotics for teaching computing, science, and engineering at a distance. Proceedings of the 5th IASTED International Conference on Computers and Advanced Technology in Education (CATE), Cancun, Mexico pp. 154-159 IASTED.

[5] Price, B., Richards, M., Petre, M., Hirst, A., Johnson, J. (2003) Developing Robotics e-teaching for Teamwork. The International Journal of Continuing Engineering Education and Lifelong Learning, vol. 13 issue 1-2 pp. 190-205 Inderscience.

[6] Scratch at MIT http://scratch.mit.edu/

[7] David J. Malan, Henry H. Leitner.(2007) Scratch for budding computer scientists, Proceedings of the 38th SIGCSE technical symposium on Computer science education, March 07-11, 2007, Covington, Kentucky, USA.

[8] John H. Maloney , Kylie Peppler , Yasmin Kafai , Mitchel Resnick , Natalie Rusk, Programming by choice: urban youth learning programming with scratch, Proceedings of the 39th SIGCSE technical symposium on Computer science education, March 12-15, 2008, Portland, OR, USA.

[9] Phidgets http://www.phidgets.com/

[10] PicoBoard (ScratchBoard) http://www.picocricket.com/picoboard.html

[11] Open University OpenLearn http://openlearn.open.ac.uk/

Public Digital Note-Taking in Lectures

Roshni Malani
UC San Diego
9500 Gilman Drive, MC 0404
La Jolla, CA 92093-0404 USA
rmalani@cs.ucsd.edu

William G. Griswold
UC San Diego
9500 Gilman Drive, MC 0404
La Jolla, CA 92093-0404 USA
wgg@cs.ucsd.edu

Beth Simon
UC San Diego
9500 Gilman Drive, MC 0404
La Jolla, CA 92093-0404 USA
bsimon@cs.ucsd.edu

UbiComp 2009, Sep 30 – Oct 3, 2009, Orlando, FL, USA

Abstract

Note-taking during lectures is a predominant activity among students. Making notes *public* allows all students to benefit from solitary actions, while making notes *digital* minimizes cost of sharing. We hypothesize that *public digital notes* (1) do not significantly change ingrained note-taking practices and existing classroom dynamics, (2) support pedagogical practices, and (3) consider the student perspective. Public digital notes are democratic in nature and motivating to students. We explore the breadth of this design space with three different projects. (1) NoteBlogs are notes taken by a few self-selected students using Tablet PCs on top of instructor prepared slides. These notes are shared instantaneously during lecture. (2) Collaborative SearchNotes bring outside resources into the lecture. All students can search for lecture terms on the Web and view their peers' findings. (3) Integrative Notes are written with a digital pen on digital paper and imitate traditional student note-taking as closely as possible. This project explores the benefits of superimposed versus juxtaposed notes shared publicly after lecture. User studies of each of these projects suggest that some students will produce content, while the majority will consume the content. Yet, we find that the process of sharing is beneficial to both producers and consumers, whether as a means of explanation, self-expression, or reassurance.

Keywords
Student Note-Taking, Lecture, NoteBlogs, SearchNotes, Integrative Notes, Ubiquitous Presenter

ACM Classification Keywords
K.3.1 [Computers and Education]: Computer Uses in Education – collaborative learning.

Introduction

The practice of writing information on paper while listening to lecture is universally perceived as an important skill for students, the responsibility of students, and the key to academic success [3]. Faculty and students alike hold this perception [9]. Note-taking serves two main functions: (1) the process of encoding information during lecture and (2) the product that stores information for later review [7]. Although student note-taking is such a ubiquitous activity, most students are not explicitly taught how to take notes. A survey of 223 University of Georgia students revealed that 99% of them took lecture notes but only 17% had received any formal instruction in note-taking [15]. As a result, students generally capture a relatively small percentage of critical lecture ideas [5,13]. Salient theories of cognition and learning [4,8,18] indicate that many *pedagogical opportunities* exist for improving the metacognitive skills of student note-takers by involving the community in the note-taking activity and by encouraging the social construction of knowledge.

The design of Web-enabled note-taking technologies must support capture during lecture and access after lecture. Orthogonal functions of manipulation (adding, deleting, editing, organizing, and searching notes) and collaboration across time and space must also be supported. The design of these technologies face some *challenges*: (1) ubiquitous, inexpensive hardware technologies must be used, so that all students can afford to use it, (2) the technology must be simple to learn and easy to use, because students are already cognitively overloaded with the task of comprehending new lecture material, and (3) technologies that involve the aggregation of student-generated content must scale well.

The pedagogical opportunities inherent in extant note-taking practices and the technological design challenges motivate what we call *public digital note-taking*. *Digital* notes, that is, notes recorded using a computing device, such as a Tablet PC, a PDA, or a digital pen, provide affordances, such as editing, reorganizing, searching, and copying, and enable sharing of individual notes with little overhead. *Public* notes may benefit all students in the classroom by using networked technologies to share notes. We hypothesize that these notes should be embedded in lecture materials to minimize distraction and may extend beyond the content presented during lecture to provide a broader context for learning.

In this paper, we define the design space for public digital notes, review prior work, and present three projects that explore this space. Data gathered from deploying each of these projects in live lectures is presented, followed by a comparative analysis focused on changes to prevailing practices and pedagogical benefits. We conclude with some design guidelines and implications for instruction.

Design Space

The design space of public digital notes consists of at least the following four major dimensions: (1) form

factor, (2) time of sharing relative to lecture, (3) percentage of students generating content, and (4) direction of information flow.

The form factor of public digital notes includes the physical size, shape, and weight of the hardware technology used, and power consumption and network connectivity. For example, a Tablet PC in comparison to a digital pen consumes a lot more power and is significantly heavier, but provides more computational power and wireless network access. A more generic laptop might tend to be smaller, but does not provide the affordances of a handwritten interface. Form factor is a very important dimension to consider because most students often carry the technology all day and use it in the confined space of modern lecture chairs, which are bolted in neat rows, have small desks, and limited or no access to electrical outlets.

When the notes are shared relative to lecture is another design dimension of public digital notes. Most notes, with some amount of effort, can be shared after lecture. For example, notes captured with a digital pen can easily be transferred to a computer connected to the Internet. In contrast, technologies that provide easy access to wireless networks during lecture, such as laptops and Tablet PCs, can enable the instantaneous sharing of notes. The time of sharing, that is, when the digital notes are made public, may influence the content and nature of the notes themselves.

The percentage of students in lecture who are generating public digital notes can vary. This percentage depends on the affordability and ubiquity of the note-taking platform, as well as the nature of notes. For example, small Post-It-sized notes generated by all students may be more feasible to review compared to lengthy freeform notes from all students. Additionally, the relative percentage of students generating content affects the design of how software aggregates and presents the public digital notes.

The final design dimension to consider is the direction of information flow relative to lecture. Traditional note-taking involves the transfer of information from inside the classroom to outside: students capture content during lecture and review it later outside of the classroom. Another possible flow of information may occur from outside the lecture to inside, for example, when students read the textbook prior to lecture and reference it during lecture. Furthermore, information can flow from student to student within lecture, for instance, when students communicate with each other during lecture.

Prior Work

Research on technologies that explicitly support note-taking activities in lectures has focused on providing notes that are either superimposed on or juxtaposed with prepared lecture slides. Most of these technologies have focused on individual note-taking, some have explored taking small Post-It sized notes, and a few have supported small group note-taking. For example, eClass and Classroom Presenter support individual student annotation superimposed on lecture slides [1,2]. LiveNotes allows each student to take their own superimposed notes, while concurrently viewing the annotations of a small group of their peers [12]. This system provides no explicit division of labor or management of space conflicts. NotePals enables students to take small notes during lecture. Afterwards

these notes are juxtaposed with the lecture slides and are shared with the entire class [6]. In contrast to LiveNotes, students using NotePals are unaware during lecture of other students' notes, which minimizes space and content conflicts but may result in duplicated effort.

Interesting research has also been done on different types of note-taking activities, such as while reading or brainstorming, and technologies to support these activities can be adapted easily into the ecology of traditional lectures. Papiercraft involves pigtail gestures on physical paper to indicate computing commands (such as copy, paste, link, search, and email), which are later executed on the digital version of the document [14], whereas InkSeine focuses on immediate feedback for in situ search for related material [10]. In contrast to these systems for individual note-taking, GroupScribbles provides explicit mediation of collaborative exercises. This system is based on the metaphor of each student holding a pad of Post-It notes and contributing thoughts to a shared whiteboard [16]. In addition, Steve Whittaker's work on social summaries (using handwritten annotations and photos to tag different parts of lecture recordings) found that students prefer social tagging systems and rely on popular tags [11].

Most of these systems support individual note-taking in novel ways. Only NotePals supports *public* notes, which are limited in size and shared after lecture. What happens when notes are shared during lecture? How can technology encourage students to gather and share different resources during lecture? What is the tradeoff between affordable, lightweight technology and the time at which notes are shared?

System Implementation

NoteBlogs are primarily based on the concept of blogs, which are ongoing narratives or personal diaries that are published on the Web. Blogs provide a medium for communicating thoughts and feelings, and a forum for reflecting on experiences. We recognize that traditional notes can similarly be viewed as ongoing personal narratives and that pedagogical benefits may emerge from publishing these notes instantaneously on the Web. The NoteBlogging application, designed for a Tablet PC, enables a small percentage of self-selected students to share notes taken on top of instructor prepared slides immediately. These notes are shared live during lecture, and all students can view them during class, creating a flow of information amongst students within lecture. Thus, all students can benefit from the thoughts and reflections of a few students.

Collaborative SearchNotes are based on the metaphor of collecting relevant resources in a shared place when investigating a new topic. This technology aims to integrate resources that are easily accessible and searchable on the Internet as part of the lecture content. All students can cooperate to find and share relevant resources. Collaborative SearchNotes enables all students who have a laptop or other Web-enabled device to bring outside resources into lecture.

Finally, *Integrative Notes* aims to mimic traditional student note-taking as closely as possible, using the familiar form factor of digital pen and paper (using the Livescribe technology). Individual notes taken by self-selected students are shared after class in the context of lecture slides. This technology explores the tradeoff between form factor and time of sharing.

NoteBlogs Data

In-depth information about the implementation, deployment, data gathering and analysis of NoteBlogs is provided in [17]. The system was used in a quarter long introductory computer science course, with a mid-quarter reelection of bloggers. From semi-structured interviews with the students, we evidenced changes in student behavior and perception of note-taking. The bloggers clearly had an audience in mind, because they "tr[ied] to make things clearer for other people" [B1]. Bloggers provided alternative explanations, because "people don't want to look at the same exact thing for each blogger, they want to read different stuff" [B2] Bloggers tried to give hints or suggestions for solving in-class active learning exercise. As B1 explained, "some of the in-class problems, like, if you're completely new to computer science, like it would take you way longer than the professor gave you time for in order to solve the problem, [so] if, just the few hints of from, like, where to start, like, what to focus on, could help them write the program a little faster, if they choose to read it." Bloggers valued NoteBlogs as a means of self-expression and communication, because "it's just like a cool way to get your opinion out there for everyone to see" [B2].

Students reading the blogs (called watchers) sought assistance and reassurance in the blogs. One watcher found that "they won't like give you like the answer directly, as much as they'll [...] give you [...] hints or like how to solve it or the logic behind like solving it" [W1]. Another watcher explained that "when [he tries] to solve a problem, [he's] not sure if [he's] doing it right, not sure if the blogger is either, but it gives you a feeling that you're probably going the right way" [W2].

Collaborative SearchNotes Data

An example of searching a term and viewing the results in the context of lecture is demonstrated in Figures 1 and 2. In a semester long modern physics course with 15 students, 36% of the search actions were searching new terms, 53% were viewing search results, and 11% were deleting automatically saved search tabs. Of these search actions, 86% were performed during lecture. As shown in Figure 3, over 80% of the search queries were on topic and related to the course and 20% of the students created 51% of the new search queries.

In a quarter-long social research methods class, semi-structured interviews with the students indicated that they had many reasons to search the Web for class related content. One student reported searching "terms that [she] didn't quite understand" and "terms [that] are really similar to each other, like the different types of sampling" [S2]. Another student used the system "just to see if on the Internet it's explained differently" [S1]. This student acknowledged that "the book is more related to the class," but the Internet provides "another perspective [...] without any bias" [S1]. Students recognized that searching online can be faster, especially "if things in the book were spread out and weren't clear" [S5].

Did students value the automatic saving and sharing of search results? A student in the physics course explained why he looks at what others have searched: "It's quite possible that they are searching for

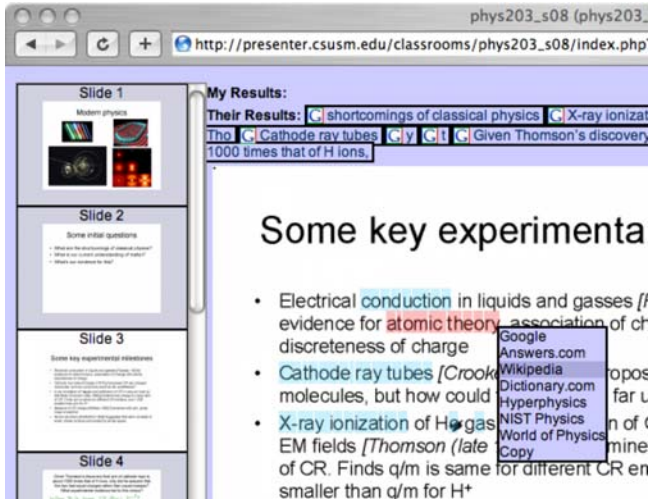

Figure 1. Example of searching a term.

Figure 2. Example of viewing a search result.

something I had not thought of and this would be information that I may not have ever found out about" [S4]. A student in the sociology course said that "usually [she ends] up using other people's searches because they usually beat [her] to it" [S2]. Students also found reassurance in reviewing their peers' searches. As S1 said, "It is helpful to see what [my peers have] done too, because I can see I need that term too and it makes me feel better that they looked it up too cause it's not just me that didn't understand."

These interviews also revealed that Collaborative SearchNotes appeals to students who are already confident in their search abilities, while encouraging others to engage in this novel, discovery-based activity. One student explained, "I've always been the kind of person that will look up stuff outside of class if I'm interested in it or if I didn't understand it well enough. I mean I will go home and I'll search it or I'll ask other people about it. So I mean that's something I've always done" [S2]. This student indicated that she doesn't "generally bring [her] computer in to class ever," but she does bring her laptop "just for this class for the purpose of using the program." This self-reported claim is confirmed by in situ observations during the course.

Taking SearchNotes was an emergent behavior amongst a few students. As one student explains, "Just for this class, I do search. I started to look for [terms], because it's really easy to access. I never thought of looking up terms on the Internet, you know. It's just like you read your notes and that's it. But then, I'm like oh, you can see what Wikipedia or Google says. I never thought of looking for extra information" [S1]. This student concluded that SearchNotes "really helped,

because it gives you another perspective of [the lecture]."

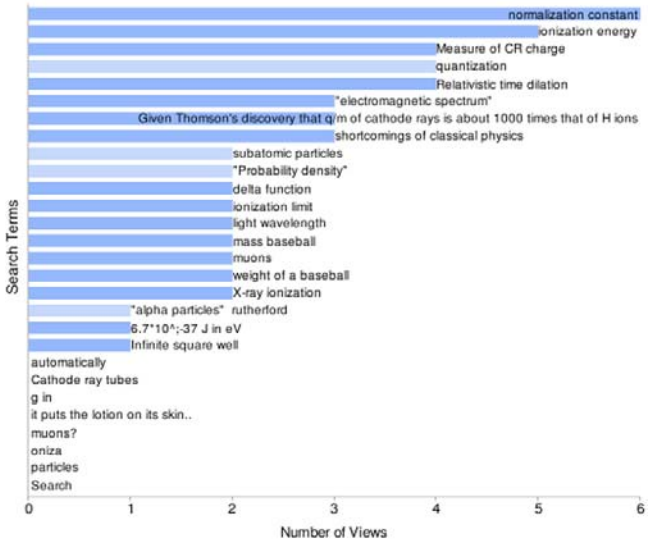

Figure 3. SearchNotes Content.

Observations of the system during lecture indicated that the current SearchNotes implementation limited students' ability to express themselves. Students could choose and refine the search terms, but they had no place to record reflections on what they discovered from a given Website. We were reminded that students value the opportunity to paraphrase what they are hearing and reading, as a way to reinforce their own understanding. Also, a slow rate of adoption of SearchNotes was observed during lecture. The purpose of the work was not to revolutionize the note-taking of all students, but rather to leverage the search efforts of a few students to benefit their peers. The data indicates that instructors and researchers should model good search behavior, and that sometimes a small incentive, such as a few extra points, can make the long-term benefits of inquiry-driven learning appear more immediate.

Integrative Notes Data

An example of Integrative Notes juxtaposed to the lecture slides is shown in Figure 4. This system was used in two different quarters of the same introductory programming course. Both versions of the course had approximately 50 students, and two students volunteered to take Integrative notes in the first quarter and four in the following quarter.

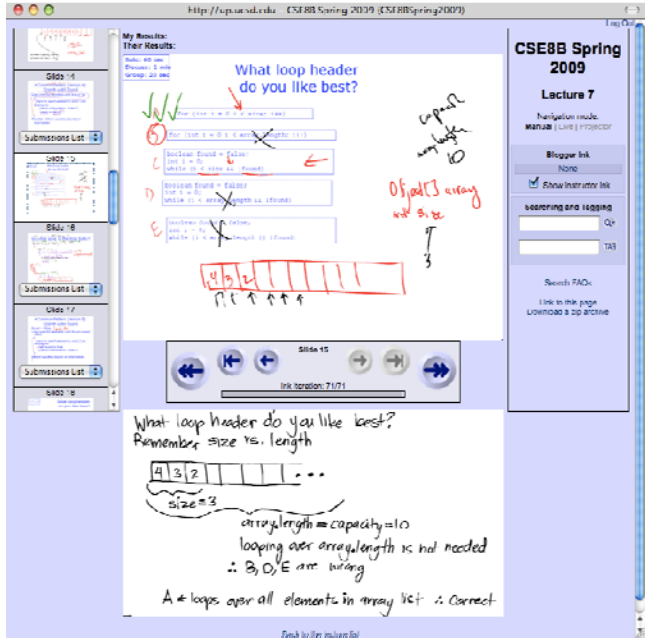

Figure 4. Example of juxtaposed Integrative Notes.

Interviews

When asked how do you feel participating in this experiment benefits you, one Integrative note-taker responded, “I feel that since others are going to look at them, I should take good notes which makes it easier for me to study, because my notes are that much better” [I2]. Thus, the social pressures of performing well in front of others enhanced the quality of his notes, which in turn improved his understanding of the material. On the other hand, another student claimed that “I wouldn't say it helped ME more, (maybe subconsciously), but I hope it helped others” [I1] (emphasis not mine). Interestingly, this student professed that he usually does not take notes: “I usually don't take good enough notes to the point where other people can understand them, and on top of that I didn't normally take notes in [this] class, because we were encouraged to bring our computers and participate thru clickers or on UP.” Despite his natural inclination not to take notes in this class and his worries about “how atrocious [his] handwriting could be,” this student continued to participate in the experiment. Another student who volunteered to participate in the experiment only took notes for one lecture, and then declined to continue participating for similar reasons. He told us that he likes the idea, but would rather spend lecture time discussing verbally with his classmates during the active learning exercises.

Previously, in the NoteBlogging study, we found that students are aware consciously of an audience of other students who would be reading their notes. Is the pressure of other students reading your writing diminished when the sharing is not live but delayed until after the lecture? We asked the Integrative notetakers whether they are consciously aware of an audience when writing notes or if they simply continue to write notes as they normally would. One student responded, “Yes, I consciously took better notes because I knew people would be reading them,” adding that “I tried to write/illustrate concepts in a different way from the teacher to give people a different perspective” [I1]. Another student said that, “At first I was just taking notes for myself, things that I thought I need to know, then when I saw that my notes were possibly going to be shown to everyone, I started taking more precise notes, and things that I thought would help them, because it helped me out” [I2]. Thus, sharing after lecture only delays the realization of an audience, but does not eliminate it over the duration of the course.

In this study, we also found that the Integrative note-takers appreciated having other students share their notes as well. For one student, it alleviates some of the performance anxiety: “I think its good to have more than one person share notes, I wouldn't want all that pressure" [I1]. This student also made an interesting point about number of students sharing notes: “it would be like being a TA if I were the only one providing notes.” Another student modestly recommended that “I just want the best notes to be shown, whether they are mine or one of the other note takers, it does not matter” [I2]. Thus, students explicitly recognize that they want high quality Integrative notes.

Are there any costs to the Integrative note-takers in sharing their notes with all the students in the course? The student concerned about his handwriting conjectured that “maybe having to write more legibly

made me miss things" [I1]. Another student speculated that "the only thing that I imagine losing is shorthand, but even then I don't use it often enough for it to be a big deal" [I2]. We asked these students whether they feel that other students (especially the ones that didn't show up to class) benefitted from the sharing of their notes more than they did. One student replied, "perhaps they might have, but it's not something that really bothered me. I'm not competitive to the point of avoiding helping a classmate understand (plus it just solidifies my knowledge when I do)" [I1]. Another student's response is unexpectedly positive: "I would hope so, I try to write notes as though someone that does not know anything is going to read my notes. This makes it easier for me to understand the notes to a better degree, and also come back to it later on" [I2]. This student states that "If someone needed my notes, they I would give it to them. I just want everyone to succeed". Thus, this evidence suggests that some computer science students are not competitive and willing to help their fellow peers.

In situ Observations

Observations of the Integrative note-takers indicate that they are avid note-takers, often writing notes even if no other student in the class is. For example, both Integrative note-takers were the only students writing notes when the concept of enumerations was introduced. These students are often writing when the instructor is explaining the solution to an active learning exercise, and especially so when the student answered the question incorrectly. The student who takes Integrative notes superimposed on the lecture slides is often shifting his gaze between the projected slide and his paper notes: "R3C7: taking notes (look back and forth between paper & slide), Prof asked ? & he attended to her, clicked answer & now taking more notes". Also, these self-selected students are responsible for their own learning, and they bring additional resources to the lecture: "R2C4: looking through book to find answer to clkr ? ".

The most interesting observation is how the two most active Integrative note-takers would frequently tag-team to make sure all the important content was captured. For example, one day in lecture, "R2C4: Took break from notes & looked at R3C7 while Prof started talking really fast" and then "R3C7-C8: wrote notes like crazy about new concept". Another day, at one point: "R3C7: taking notes on what [Prof's] saying, R2C4: wasn't taking notes while [R3C7] was → Now both taking notes" and then they switch at another point: "R2C4: Taking notes R3C7 wasn't, but wrote at end of Prof narrative - Both stop for new [clicker] ? ".

Surveys

All the students in the course were asked to fill out a short survey. The first two questions on the survey asked students whether they used the Integrative Notes system to prepare for the exam and how much of the handwritten student notes they read. Of the 111 students total that responded to the survey, a total of 55 students used the system and read any of the Integrative Notes. Approximately 20% of the students did not use the system to study before the exam. In the open response portion, one student reported "I downloaded the slides" and another student said "I'll check them out next time, I was too busy this time." Some students were not aware of the availability of student-written notes on the UP website, despite repeated announcements and demonstrations during

lecture. One student commented “We have student notes on UP!?” on the survey.

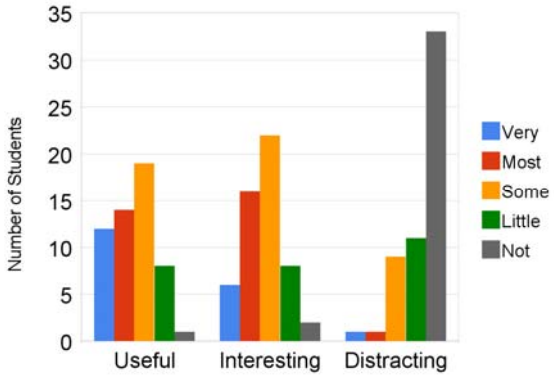

Figure 5. Integrative Notes Survey Summary.

The next three questions of the survey asked students about how useful, how interesting, and how distracting the handwritten student notes were. The answers to these questions, given by students who read any of the notes, are summarized in Figure 5. More than 80% of the students reported that Integrative Notes were somewhat, mostly or very useful, and a similar percentage reported that the notes were interesting. 22% of the students found the notes to be very useful. 60% of the students indicated that the Integrative Notes were not distracting at all.

The general feedback varied from positive responses such as “cool idea” and “very useful, especially [the] feature that allows me to toggle instructor ink on/off” to mixed responses such as “student notes wasn't that bad” and “the notes were not much different from what was covered in lecture.” Students recognized that the Integrative Notes were a “good supplement” because “it came in handy when I couldn't read [the instructor's] slides” and because the Integrative note-takers “were able to write down the important things said in lecture that would take too long for professor to write.” Also, one student indicated a learning curve: “I just need to get more familiar with it and I will take more advantage of it.”

Changes to Prevailing Practices

Collaborative SearchNotes was the most radical project, in the sense that even if students are accustomed to having access to Web-enabled devices during lecture, they do not perceive gathering resources as a form of note-taking. Students highly value paraphrasing, summarizing, and expressing new ideas in their own words, because this encoding process reinforces their learning. Even though SearchNotes greatly simplified and empowered the storage function of note-taking by automatically archiving resources in the context of lecture, it did not provide much room for encoding and self-expression. Students were able to select the query terms and phrases, but had no opportunity to reflect on what they found. No students abandoned their own note-taking (either on paper or electronically) to rely solely upon SearchNotes.

In contrast, Integrative Notes was the most similar to conventional note-taking, in that the form factor was natural to all students and the learning curve was negligible. The interface for taking notes during lecture (pen and paper), the interface for sharing notes after lecture (USB dock and email), and the interface for viewing the shared notes (Ubiquitous Presenter website) were all different, but each individual interface was familiar to students. Due to the disconnect in the

interfaces and the delay between the time notes were written and shared, the realization of the consequences of shared notes was not as evident to some of the students. Thus, some students did not significantly change their note-taking practices, while those that changed their practices focused on providing clear, precise explanations and illustrating concepts in different ways.

Changes required by NoteBlogging lie somewhere between these two extremes. The form factor is mostly familiar: a laptop with a handwriting interface. However, using a pen to navigate the functionalities of the computer (like pressing command or shift keys) and to write (finding the correct angle and pressure, and becoming comfortable with resting a hand on the screen) take a little while to adjust to. After that initial learning curve, the note-taking also differs in that, as a student is writing notes superimposed on the prepared slides, the instructor's annotations also start showing up, sometimes causing space conflicts. Most bloggers resolved this by waiting until the instructor had moved on to the next slide before writing their comments. Finally, since the notes were shared live during lecture, bloggers were aware of an audience of watchers and strove to provide helpful hints and suggestions during active learning exercises.

Pedagogical Benefits

The pedagogical benefits of all three forms of digital note-taking arose primarily from the public aspect. Shared notes allowed students to see what their peers thought was most important during lecture, and encouraged them to learn from each other. Without explicitly being told to do anything different than they normally would during lecture, students determined how to maximize new utility given the new affordances of the technologies.

In Collaborative SearchNotes, many students reviewed the pertinent search results of only a few other students. A few students felt confident in their abilities to search for related content on the Web, and thus, engaged in exploratory and inquiry-based learning. Their selfish actions had a side effect of introducing new material and different explanations to other students and indirectly teaching their peers how to formulate good search queries.

In Integrative Notes, students focused on providing clarity and organization to the material covered in lecture. The most common behavior was to capture the verbal explanations of the active learning exercise given by the instructor. However, since different students paid attention to different aspects of what the instructor said and since they have different prior knowledge, students often wrote different things. One student decided to structure her juxtaposed notes in an outline format, highlighting at a quick glance all of the important concepts covered during that lecture.

NoteBloggers clarified, organized, and explained differently the concepts presented in lecture, and also provided hints and suggestions to active learning exercises live during lecture. The affordances of immediate sharing allowed watchers to attempt to solve an active learning exercise, where they might not have been able to otherwise.

In addition to motivating students to generate public digital notes, all three systems created interesting, class-related content for other students to consider.

Rather than forgoing an in-class problem solving exercise or becoming distracted, students had a resource such as NoteBlogs to check for hints and suggestions from their fellow students. Rather than opening a new tab to check email or read RSS feeds, students using SearchNotes had search tabs along the top of the lecture slides to read first. If students became frustrated with illegible instructor ink gestures, students could check if their peer bloggers or Integrative note-takers had explained the same concept in a similar manner. All students found assistance and reassurance in the public digital notes.

The addition of public digital notes to the lecture material did not distract the students. In Integrative Notes, 60% of the students in an introductory programming course reported not being distracted by the posting of individual notes of a few students to the lecture slides, where as more than 80% found the public notes to be useful and interesting. Over 80% of the search queries were related to the course, and thus automatically saving and sharing the search results was not distracting to the students. More than 90% of the noteblog content in an introductory programming course was about programming, and even expressions of mental state or empathy were reassuring to the watchers.

Conclusion

Note-taking during lectures is a pervasive practice amongst university students. The objective of this work has been to exploit the solitary practice of traditional note-taking to benefit mutually all students in the course without significantly altering the learning ecology. We have demonstrated how notes taken solitarily can be shared with all the students in the context of lecture materials. In particular, we built, deployed, and evaluated three different systems to facilitate public digital note-taking: one based on the metaphor of blogging, another guided by the idea of incorporating lecture-related resources found on the Web, and a third exploring the tradeoff between form factor and time of sharing.

These three projects span the breadth of the public digital note-taking design space, exploring aspects of communication and sharing that have not been researched extensively yet. Figure 6 presents the communication aspect of public digital note-taking. The horizontal axis is how many students are producing, or taking, notes and the vertical axis is how many students are consuming, or reviewing, those notes. As the figure demonstrates, the projects presented in this work span the space of public notes, that is, the space where all students in the class consume and review the notes. Furthermore, all three projects use different form factors, which in turn, affects the number of students who can produce notes.

Figure 7 presents the sharing aspect of public digital note-taking. The horizontal axis represents when the notes are made available for sharing, whether delayed until after lecture or live during lecture. The vertical axis depicts whether the collaboration around the shared notes is explicit or implicit. Explicit collaboration is when the notes of one student directly affect the content of another student's notes, whereas implicit collaboration is when a student takes note of what he/she thinks is important, with little concern for the effect that note may have on other students. Most prior work has studied explicit collaboration, whereas the work here is more concerned with means of implicit

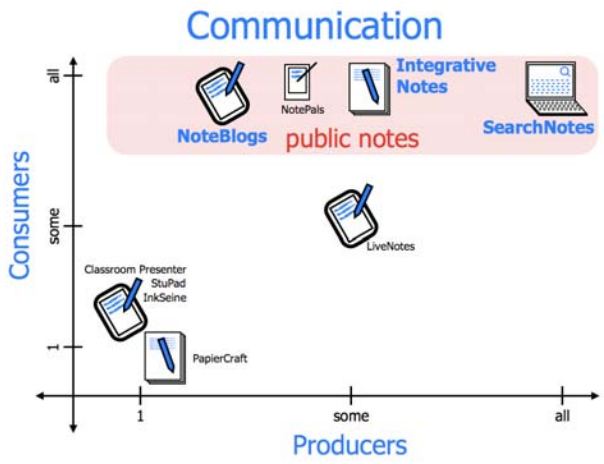

Figure 6. Communication aspect of public digital notes.

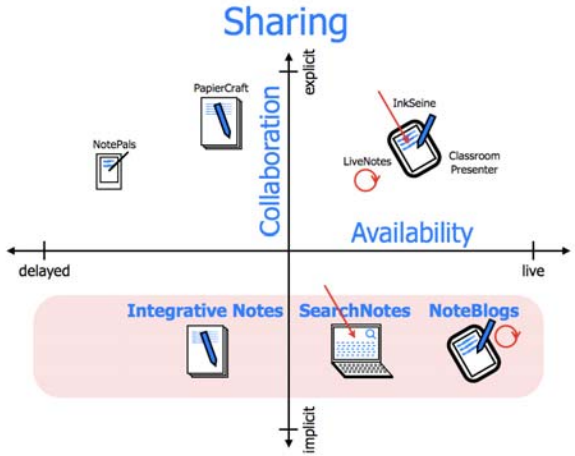

Figure 7. Sharing aspect of public digital notes.

collaboration. Another thing to note about this chart is the red straight and circular arrows. In traditional note-taking, information generally flows from within lecture where it is captured and stored to outside of lecture for later review. In contrast, InkSeine and SearchNotes emphasize and encourage the opposite flow as well, that is, the bringing in of information from outside lecture to within lecture. LiveNotes and NoteBlogs allow students to share their own interpretation of lecture material, creating a new circular flow of information amongst students during lecture.

Design Guidelines

1. *Minimize perceived changes to existing practices.* Students are already cognitively overloaded during lecture, and students should perceive the introduction of new technologies into this constrained environment as not significantly changing their natural behavior. The interactions supported by the interface should be familiar and easy to learn. For example, students are already familiar with formulating search queries and finding relevant content on the Web and viewing these results as tabs in their browser. SearchNotes simply brought this familiar interaction with one click into the lecture notes, while seamlessly adding automatic archiving and sharing.

2. *Support choice in note-taking styles.* Students have many different learning styles and note-taking habits. Some with fast handwriting may have developed a strong sense of paraphrasing and organizing, while others with slower handwriting might rely on providing clarifications and other useful annotations on top of prepared lecture slides. As Integrative note-taking demonstrated, both juxtaposed and superimposed notes are valuable resources when shared. NoteBlogs

appealed to one student who often brought many colored pens to lecture and was accustomed to using color to disambiguate and organize her notes.

3. *Provide outlets for student self-expression.* Students like to express their own thoughts in their own words. Collaborative SearchNotes limited this expression, with only the ability to express search terms, and ultimately was not as valuable to the students as the other forms of public digital notes. Expressions of self, even something as trivial as an expression of hunger during a lunch-time course or as mundane as an admission of making the common novice mistake, are reassuring and valuable to peers who can commiserate.

Implications for Instruction

1. *Encourage voluntary participation.* The process of self-selection automatically filters students to match their interest and abilities with the technology. For example, only those students who felt confident in their search abilities produced SearchNotes, and those students who were comfortable trying new note-taking methods volunteered to NoteBlog and to write Integrative Notes. Both NoteBloggers and Integrative note-takers felt a sense of social responsibility to their peers. All three public digital note-taking tools have the potential to engage high performing students, taking advantage of this selection bias to aid all students.

2. *Select more than one student to participate.* From the pool of volunteers, select a small group of students to participate in public digital notes. As one Integrative note-taker said, there is too much pressure to be thorough and correct if he is the only one sharing notes. A small group of students sharing allows everyone to benefit, including those generating content. The more students generating public digital notes, the greater the variety of viewpoints and alternate explanations. For example, the bloggers in one course clarified the difference between `static` and `final` keywords in Java in two distinct yet complementary ways. Similarly, of the three Integrative note-takers in another course, one explicitly defined the terms `this` and `super` in a constructor, another discussed default behavior, and a third explained the debugging advantages.

3. *Offer immediate rewards to help students achieve long-term pedagogical benefits.* Students may not recognize the long-term benefits of peer learning and inquiry-based learning, especially if the note-taking tool is perceived as altering current practices. Incentives, such as a negligible amount of bonus points toward their grades or contests, motivate students to try the novel application for perhaps enough time to get accustomed to and eventually adopt it. In the NoteBlog study, a re-election of bloggers halfway through the course bolstered spirit and pride in the bloggers and encouraged excellent note-taking.

Thus, we find that students are more likely to embrace technologies that they perceive as minimally changing their existing practices, such as Integrative Notes and NoteBlogs. Even though students perceived minimal changes, these systems enabled peer learning and enhanced the sense of community amongst students. Those generating public digital notes endeavored to add clarifications, organizational structure, and alternative explanations, while most students found the shared notes to be useful, interesting, reassuring, and not distracting. Appealing to student's intrinsic motivation is essential to this work. Students feel a sense of social responsibility for their note-taking work

and are rewarded with recognition amongst their peers. The challenge for public digital note-taking systems is to maximize student choice and control and provide some feedback mechanisms that scale.

Acknowledgements

This work was supported by equipment donations from Hewlett-Packard and Livescribe, by a gift from Microsoft Research External Research and Programs (MSR ER&P), and by a NSF CCLI grant (DUE-0618511). We thank the instructors and students who participated.

References

[1] Anderson, R., McDowell L., and Simon, B. Use of classroom presenter in engineering courses. In *ASEE/IEEE Frontiers of Education 2005*, T2G:13-18.

[2] Brotherton, J.A. and Abowd, G.D. Lessons Learned From eClass: Accessing Automated Capture and Access in the Classroom. In *TOCHI 2004*, ACM Press, 121-155.

[3] Carrier, C.A., Williams, M.D., and Dalgaard, B.R. College students' perceptions of notetaking and their relationship to selected learner characteristics and course achievement. *Research in Higher Education*, 28(3):223–239, 1988.

[4] Cole, M. *Cultural Psychology: A Once and Future Discipline*. Harvard University Press, 1996.

[5] Crawford, C.C. The correlation between college lecture notes and quiz papers. *Journal of Educational Research*, 12(4):282–291, 1925.

[6] Davis, R.C., Landay, J.A., Chen, V., Huang, J., Lee, R.B., Li, F.C., Lin, J., Morrey, C.B., Schleimer, B., Price, M.N., and Schilit, B.N. Notepals: lightweight note sharing by the group, for the group. In *CHI 1999*, ACM Press, 338–345.

[7] DiVesta F.J. and Gray, G.S. Listening and note taking. *Journal of Educational Psychology*, 63(1):8–14, 1972.

[8] Engeström, Y. *Learning by expanding: An activity-theoretical approach to developmental research*. Orienta-Konsultit Oy, Helsinki, Finland, 1987.

[9] Hartley J. and Davies, I.K. Note-taking: A critical review. *Programmed Learning and Educational Technology*, 15(3):207–224, 1978.

[10] Hinckley, K., Zhao, S., Sarin, R., Baudisch, P., Cutrell, E., Shilman, M., and Tan, D.. Inkseine: In situ search for active note taking. In *CHI 2007*, ACM Press, 251–260.

[11] Kalnikaité, V. and Whittaker, S. Social Summarization: Does Social Feedback Improve Access to Speech Data? In *CSCW 2008*, ACM Press, 42-51.

[12] Kam, M., Wang, J., Iles, A., Tse, E., Chiu, J., Glaser, D., Tarshish, O., and Canny, J. Livenotes: a system for cooperative and augmented note-taking in lectures. In *CHI 2005*, ACM Press, 531–540.

[13] Kiewra, K.A. Notetaking and review: the research and its implications. *Instructional Science*, 16(3):233–249, 1987.

[14] Liao, C., Guimbretière, F., and Hinckley, K. Papiercraft: a command system for interactive paper. In *UIST 2005*, ACM Press, 241–244.

[15] Palmatier, R.A. and Bennett, J.M. Notetaking habits of college students. *Journal of Reading*, 18(3):215–218, 1974.

[16] Roschelle, J., Tatar, D., Chaudhury, S.R., Dimitriadis, Y., Patton, C., and DiGiano, C. Ink, improvisation, and interactive engagement: Learning with tablets. *Computer*, 40(9):42–48, 2007.

[17] Simon, B., Davis, K.M., Griswold, W.G., Kelly, M., and Malani, R. Noteblogging: taking note taking public. In *SIGCSE 2008*, ACM Press, 417–421.

[18] Vygotsky, L.S. *The Collected Works of L. S. Vygotsky: Problems of General Psychology*, Volume 1. Plenum Press, New York, second edition, 1987.