

Using Podcasts and Tablet PCs in Computer Science

Barry L. Kurtz, James B. Fenwick Jr., and Christopher C. Ellsworth

Appalachian State University

Boone, NC 28608

{blk,jbf,cce}@cs.appstate.edu

ABSTRACT

This experiment involved our software engineering course that is required for students majoring in Computer Science; however our experiences are applicable to most computer science courses. The Fall 2005 course was taught based on slide presentations during class and work on the course project outside of class. We inverted this arrangement during the Spring 2006 course: lectures were delivered as video podcasts outside class while class time was spent working on the course project and problem solving using Tablet PCs. The same textbook and slide presentations were used; exams were almost identical and the projects were similar. We discuss the conversion of an entire semester's worth of lectures into 65 video podcasts. About 30% of each podcast includes live video of students interacting with the instructor. We also discuss use of Tablet PCs during the class sessions. The students in both semesters had equivalent performance on exams, but the project grades in the Spring semester were substantially higher resulting in higher overall grades. We conclude with a description of our current efforts using podcasts and tablet PCs in a data structures course. This work was supported partially by NSF grant DUE 0341506 [10].

Categories and Subject Descriptors

K.3 [Computing Milieux]: Computer and Education – K.3.1 *computer uses in education*, K.3.2 *computer science education*

General Terms Human Factors

Keywords Computer science education, podcast, tablet PC

1. Background

1.1 The Use of Podcasts

There is a growing body of literature regarding the use of iPods, particularly audio iPods, for academic purposes. Although we use the terms iPod and podcast, the technology is not product specific. Any portable video player can be used to view a video on demand broadcast (vodcast). As we discovered, a portable video player is not even needed to distribute vodcasts; many students prefer using PCs.

Racham and Zhang [12] review the technology of podcasting, its application in the academic milieu, and make suggestions for future research. The authors claim that producing podcasts is relatively easy; this is probably the case for recording the audio content of lectures for distribution. As we will show producing video podcasts that mix narrated slide presentations with live instructor-student interactions is more difficult.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACMSE 2007, March 23-24, 2007, Winston Salem, NC, USA.

Copyright 2007 ACM 978-1-59593-629-5/07/0003...\$5.00.

Duke University [5] distributed 20 GB audio iPods equipped for audio recording to over 1600 entering freshmen in August 2004. During the first year of use, 15 Fall courses and 19 Spring courses made use of the iPods. The primary uses were content dissemination, classroom recordings, field recordings, study support, and file storage/transfer. Foreign language and music courses fully integrated the use of iPods into course materials. In the 2005-2006 academic year usage increased with 42 courses in Spring 2006 reporting usage.

1.2 Use of Tablet PCs and Wireless Classrooms

Tablet PCs (TPCs) have been used for several years to teach computer science. As reported by Simon, Anderson, Hoyer, and Su [13] TPCs encourage active and collaborative learning. Using Classroom Presenter [4] or Ubiquitous Presenter [14] students can use TPCs to work both individually and collaboratively. These learning environments support simultaneous private work by students, student control of whether they want to submit or not, instructor viewing of submitted responses and selecting those to display, allowance for spontaneous activities, and saving submissions for review after the class has been completed.

Even more widespread than the use of TPCs is the development of wireless classrooms where students either bring their own laptops or use laptops or TPCs provided by the school. Campbell and Pargas [3] from Clemson University describe how many universities are requiring students to have wireless laptops and how instructors need to adapt their lectures to take advantage of this new capability. In addition to posting classroom materials online, instructors can provide animated demonstrations of concepts often using visualization as an effective teaching tool. They can establish collaborative learning experiences that often provide instant feedback. The laptops can also be used as communications devices between students and with the instructor.

Moody and Schmidt [11] at Washburn University provide a practical overview of establishing wireless classrooms. They not only discuss the educational advantages but also delve into network setup issues, network management issues, software management issues, and network security issues.

1.3 Tutored Video Instruction (TVI)

TVI is an alternative mechanism to export lectures outside of the classroom. Anderson, Dickey, and Perkins [1] report on teaching introductory programming courses at local community colleges using TVI. The lectures were recorded at the University of Washington and exported in digital format showing the slide presentations with a small insert of the instructor talking. Students attended a scheduled class session at a local community college where the lectures were played back. The local instructor can pause the lecture to ask questions or students can ask questions during the pauses. This has the advantage of allowing live student interactions with an instructor but has the disadvantage, compared with video podcasts, of requiring scheduled classroom attendance at a designated location.

2. The Course Organization

Our software engineering course (CS4667) is normally taken during the senior year; the enrollment in the Spring semester is typically larger than the enrollment in the Fall semester. The textbook is *Object-Oriented Software Engineering: Using UML, Patterns, and Java* by Bruegge and Dutoit [2]. We cover Chapters 1-2, 4-11, and 15. The materials in Chapter 8 are supplemented to include eight design patterns in addition to the nine patterns covered in the textbook. Student grades are based on exams (40%), the course project (40%), and homework, presentations, and class participation (20%).

2.1 The Fall 2005 Course

Lectures were given in class using slide presentations. The slides were based on those provided by the authors of the textbook, but many slides were removed when they did not correspond to the textbook materials and other slides were added. The first part of the course was spent going through the textbook materials to enable students to work on the course project; there was a greater emphasis on the project itself in the second part of the course. Student interaction during lectures was largely based on the instructor asking students to answer questions or to solve small problems. There were very few student questions on the slide presentations themselves.

2.2 The Spring 2006 Course

There was absolutely no lecturing from slides in class; rather, all lectures were viewed outside of class as video podcasts. There were three components in a podcast: the instructor narrating slides, three students (referred to as “classmates”) asking questions about the lecture, and the same three classmates discussing questions asked by the instructor. The latter two components were live video and constituted about 30% of the podcast. This represented far more instructor-student interaction than occurred during in-class lectures in the previous semester. Class time was spent entirely on problem solving, UML document preparation and activities related to the project. The initial UML documents were small exercises not related to the project. There were four Tablet PCs, one for each of the student groups, connected via a wireless network. Classroom Presenter and Microsoft OneNote were used to share and work collaboratively on documents. Details on the software engineering project can be found in Fenwick, et. al. [8].

3. Production of Video Podcasts

We purchased eighteen video iPods; each student in the Spring class was issued an iPod for the duration of the semester and turned it back in at the end of the semester. No iPods were lost or damaged during the semester. The iPod not only allowed students to download and view the video podcasts but it also functioned as a reasonable sized portable hard drive for software and files. We preloaded the iPods with the Java 5 programming environment, searchable Java API documentation, Eclipse IDE, Poseidon Community Edition [9], and several other useful software programs. The tablet PCs also included Visual C# 2005 Express Edition and the Visual Web Developer 2005 Express Edition.

3.1 The Software Tools

Chris Ellsworth, a CS graduate student, assisted the instructor in learning how to produce reasonable quality video podcasts in an efficient manner. All development was on PCs running Windows XP. We used three commercial grade components:

- Microsoft Windows Moviemaker 2.1 – this software was used to download the raw video of student interactions from our miniDV camcorder and produce clips in .AVI format. (Camtasia can also perform this task, but Moviemaker was faster and easier to use.)
- Camtasia Studio 3.1 – this software was used to narrate the slide presentations, to capture screen images during tutorial sessions, and to intermix the student interactions to produce a single file in .AVI format.
- Quicktime Pro -- this software was used to convert the .AVI files to .MP4 files for podcasts. (Note: the latest version of Camtasia Studio 4.0 eliminates the need of using Quicktime Pro.)

We initially tried alternative shareware products that were free but found them to be poorly documented and unreliable. We quickly concluded it was best to purchase commercial grade products that are powerful, easy to use, reliable and well documented.

3.2 The Production Process

Three students from the class were hired to help make the video sessions showing instructor-student interactions. We typically could cover two or three podcasts in a one hour recording session. By paying students a nominal amount (which totaled less than \$800.00 for all students for the entire semester) we avoided many issues associated with permissions needed to show students’ faces and identify them by first name on materials that are publicly accessible.

The instructor first had to reformat the slides from the Fall semester so that they were readable on the iPod display. This typically required 32 pt font for the title, 28 pt font for major headings, and 24 pt font for subheadings all using an Arial font. To maximize readability no design formats were used (only black text on a white background). Often “busy” slides had to be split into two slides. A few slides had to be eliminated due to complex figures. Students had online access to pdf files for the slide presentations to improve readability. These were useful for reviewing for a test without having to view the podcasts again.

Because this was a first time effort, production of the podcasts only started about a week before the podcast was released. The classmates were given copies of the slides and a list of questions the instructor would ask. The students were asked to submit a list of questions they wanted to ask the instructor. A miniDV camcorder was used to record all video clips. The camera would record the students sitting at a table discussing the answer to instructor questions. The instructor was off camera but could be heard guiding the discussion. Bloopers or times when the discussion stalled were edited out during the production process. The students were good friends outside of class and this resulted in lively and animated discussions.

3.3 A Detailed Example of a Typical Podcast

Consider the seventh podcast in chapter 5, Statechart Diagrams; an 18 min 29 sec (18:29) podcast consisting of:

- The title slide (0:23)
- Four slides introducing state charts and ending with a vending machine example (6:38)
- One video segment where classmates discuss a statechart diagram for selecting a bus stop in the course project (3:05)
- One slide assigning construction of this statechart diagram to be turned in by students for grading (0:58)
- Four slides introducing nested states and superstates (2:21)
- One video segment where classmates discuss use of superstates for selecting a bus stop (2:12)
- Two slides where the instructor discusses a complex exercise to be completed by students: a statechart diagram for a four way stop light (2:52)

The total time to produce this podcast starting with reformatting the slides was less than two hours.

3.4 The Video Podcasts

A complete table of all podcasts for the course is shown in Table 1. We quickly learned to avoid podcasts that would cover an entire 50 minute lecture; it is much more desirable to develop a sequence of shorter podcasts divided up by subtopics.

Table 1: Video Podcasts

Chap.	Podcast (MM:SS)
1	Introduction to Software Engineering (40:08)
2	What is modeling? (9:54); Software Concepts (10:06); A First Look at UML (22:45); Use Case Diagrams (12:13); Class Diagrams (26:51); Sequence Diagrams (17:30); Other UML Diagrams (25:15)
4	Introduction to Requirements Elicitation (21:57) Requirements Elicitation - Parts 1 (19:58) Requirements Elicitation - Parts 2 (15:17)
5	Introduction to Static Modeling (10:01); Class Identification (23:56); Identifying Attributes and Operations (17:10); Some General Advice (19:06); Introduction to Dynamic Modeling (9:59); More on Sequence Diagrams (7:48); Statechart Diagrams (18:29); More Dynamic Modeling (10:40); Requirements Analysis Document (26:52)
6	Introduction to System Design (21:00); System Decomposition (24:19); Layered Systems (19:52); System Architecture (30:10)
7	Addressing Design Goals, Part 1 (37:45) Addressing Design Goals, Part 2 (26:44) Addressing Design Goals, Part 3 (28:13)
8	Object Design (10:50); Reuse and Inheritance (36:08); Frameworks (11:57); Introduction to Design Patterns (22:26); plus one podcast for each of 17 design patterns (from 7:04 to 31:55)
9	Object Design Participants (8:40); Specifying Interfaces (19:58); Expressing Constraints using OCL (31:05)

10	Types of Transformations (19:03); Optimizations, Mapping Associations (23:39); Mapping Contracts to Exceptions (18:01); Mapping Object Models to Tables (23:48)
11	Overview of Testing (20:35); Types of Testing (44:56); Integration Testing (18:56); Sandwich Testing (11:16); Other Testing (25:19)
15	Overview of the Software Life Cycle (11:49); The Waterfall Model (11:05); The Spiral Model (18:00); Issue-based Development (11:33); Process Maturity and Software Metrics (21:29)

4. Use of the Tablet PCs

There was one TPC for each of the four groups in the Spring semester. The TPCs were connected via a wireless network. The last several weeks of the semester the groups could check out the TPCs between class sessions. Four students brought in their own wireless laptops so in three of the four groups there were two or more laptops available. The TPCs were used in pen mode during the first few weeks of the course when working on UML design documents; for the remainder of the course they were used in laptop mode.

Two programs were particularly valuable when working with TPCs: Classroom Presenter and OneNote. Classroom Presenter was used when the instructor prepared a slide presentation with a series of leading questions that each group had to answer individually. One such slide asked the following question:

List some activities that occur when using an automatic teller machine (ATM). For example, one activity might be "login."

Each group would send their handwritten responses to the instructor's tablet. The instructor would choose which answers to display and discuss. In this example all responses would be discussed and the students would try to reach a consensus on a common list of activities.

Students used Poseidon Community Edition to construct UML diagrams. We used Microsoft OneNote to share these diagrams between the groups. Each group was assigned a page or group of pages on a single shared document. Each group would copy and paste its diagram on a specified page in the OneNote document and every group would immediately have the updated copy of the document. This was particularly useful when each group was working on a different but related activity. For example, one group might be working on a use case diagram, another group on the associated scenarios for the use cases, a third group on a class diagram, and a fourth group on a sequence diagram. The scenarios group would need to see the use case diagram to complete the scenarios. The class diagram group would need to see the problem statement (given by the instructor) and the scenarios. The sequence diagram group would need to see the scenarios and the class diagram to complete their work. OneNote was a great tool for sharing and updating this information dynamically and simultaneously on all machines.

5. The Course Project

The principal investigators on our NSF grant met at DePaul University in summer 2005 and decided to work on an itinerary planner for a variety of ground transportation systems. The project would have a multi-tier software architecture and use web

services between the layers. For more details on this inter-university cooperation, see Fenwick, et. al. [8].

5.1 The TRIP Project – First Semester

Students in the Fall 2005 semester developed a three tier system: a data storage layer that contained route information for the AppalCART bus system, an application layer that found alternative itineraries between a starting and ending bus stop, and a presentation layer that contained the user interface. All programming was in Java. Chris Ellsworth developed a functional, reference version of the project with these web services so that one group could develop and test their code independent of the other groups.

There was a rapid prototype phase with the following limitations: the data was stored in XML files containing hash maps, the itinerary planner only found one itinerary with at most one transfer, and the presentation layer was a very simple interface with pull down menus to select starting and ending bus stops. This prototype was completed after eight weeks. The final version of the project stored data in an SQL database, allowed the user to specify a starting or ending time and date, and found several alternative itineraries that satisfied the given criteria.

5.2 The TRIP Project – Second Semester

Students in the Spring 2006 semester started with the DataStorage web service and the RoutePlanner web services developed during the Fall semester. Their first task was to use the MapPoint web service from Microsoft to allow user interaction using maps of Boone NC. Specifically, bus stops were shown as pushpins on a map. When the user placed the mouse over a pushpin, it could be selected as a starting or destination bus stop. The itineraries were also shown on maps in addition to the textual instructions. Four groups worked independently each completing these components after nine weeks.

The best components of the four solutions were combined to form a baseline version of the project. During the final five weeks the four groups worked on different extensions to the project. Two new services were added: bus tracking to show the real time position of all buses on a map and notification of an impending bus arrival via either email or SMS text messaging on a cell phone. The four components of the project were simulating the real time position for all buses (the buses do not transmit GPS data), generating the maps showing the bus positions, generating the notification alerts, and changing the user interface to take advantage of these new services. The presentation layer was programmed in Java while other components in the application layer were programmed in C# using Visual Studio. Two additional software tools were essential to promote collaboration between different groups working on the same project: SharePoint Services available in Windows Server 2003 and CVS (Concurrent Versions System).

5.3 Performance on the Project Component

The project grades were based on design documents (18% of the total) and program implementation (82% of the total). As seen in the table below, the students in the Spring 2006 semester performed uniformly better than the students in the Fall 2005 semester. The gap is less for the design documents and more for the implementation. Overall, the grades were 9.5% higher in the Spring semester.

Table 2: Project Grades

	Design	Code	Total
Fall 05	76.8	75.8	76.0
Spring 06	80.4	86.6	85.5

The project components, as described in the prior sections, were not identical for the two classes. In the instructor’s opinion, the extensions to the TRIP system implemented during the Spring semester were more challenging than the original components implemented in the Fall semester. This makes the 9.5% increase even more significant.

6. The Exam Scores and the Course Grades

The exams were very similar for the two semesters; the problems were either the same or contained slight variations. As shown in the table below, when the weighted average of all exams is calculated there is no significant difference in exam performance with the grades in the Spring semester less than one percent higher than the grades in the Fall semester.

Table 3: Exam Scores

	midterms	final	total
Fall 05	78.8	77.1	77.7
Spring 06	78.2	79.0	78.5

The exam grades shown in Table 3 were based on raw scores; the instructor scaled exam grades in each class to produce a B-average grade (82-83%). The project grades were not scaled. As seen in the table below, this resulted in higher grades in the Spring semester. The table entries are percentages based on 6 students in the Fall class and 16 students in the Spring class. In each semester one student withdrew from the class.

Table 4: Letter Grades for the Course

	A	B	C	D/F	W
Fall 05	0.0	50.0	33.3	0.0	16.7
Spring 06	12.5	56.3	25.0	0.0	6.3

7. Some Lessons We Have Learned

Embedding exercises in the podcasts was very valuable. This made sure the students did not lag too far behind in listening to the podcasts. Exercises were collected about once a week.

At first students complained they could not ask questions when listening to podcasts. This was handled by having students send email to the instructor and receive replies in less than 24 hours. Often these replies were sent to the entire class if it was a general question. Some students noted errors or items that were not clear in the podcasts and sent email to the instructor. The instructor corrected or clarified the point in question and sent the response to the entire class.

In our current work (described below) in the data structures course we have moved away from live video showing classmates discussing a problem (the “talking heads approach”) to a video showing classmates working out a problem on a tablet PC. The classmates faces are not seen but they are heard when discussing and drawing out a solution to a problem. We call this a “drawing

pad approach.” The classmates can draw pictures of data structures, develop pseudocode solutions to problems, or show a table of values when tracing an algorithm.

The instructor was surprised to learn that many (and maybe most) students elected to watch the podcasts on their computers at home and did not watch them on the iPods. This means that distributing iPods is not essential for using podcasts as an instructional technique.

Half way through the semester the instructor asked students to complete an anonymous survey where students were asked about the podcasts and what could be done to improve them. Several students openly expressed dislike for the podcasts saying they were asked to do “extra” work outside of class that should have occurred in class. Some advantages of podcasts were mentioned:

- You could watch them at a place and time most convenient
- You could pause and replay as needed
- You could print out copies of the slides for easy review

Students have a difficult time getting a group of four together outside of the classroom. When each group was working on a separate component of the same project, it was virtually impossible for all groups to meet outside of classroom. Working on the project during the class time was greatly appreciated in overcoming these problems.

Connecting the Tablet PCs via a wireless network was very successful and overcame the obstacles of “old fashioned” wired computer classrooms. The Tablet PCs were primarily used in laptop mode for program development. CVS was an invaluable tool when the four groups were working on different components of the same project. When used in tablet mode, Classroom Presenter and OneNote were very powerful tools. The major problem when using the Tablet PCs was only having one TPC for each group of four students. We have since doubled the number of TPCs and students can now work in pairs.

8. Current Work

This approach is not limited to software engineering and can be used in almost all courses in the computer science curricula. The primary author is currently using podcasts and tablet PCs in a data structures course. Instruction is organized into three components:

- Preparation before class, usually 40-50 minutes, where students view the podcasts
- In-class activities involving problem solving tasks
- After class work on traditional programming assignments of short duration (less than two weeks)

One of the in-class activities is where students submit responses to short answer questions using Ubiquitous Presenter [14], a web-based version of Classroom Presenter. These responses are anonymous and the instructor would select particular responses for discussion. This allows discussion of common misconceptions when they occur.

The other in-class activity is to write a Java method to perform a specified task and submit it for testing. This component is based on our experiences with Quiver, a Quiz Verification system [6]. Quiver was developed by Chris Ellsworth using Apache Cocoon and a variety of other software tools. After Chris graduated the system fell into disuse. Therefore we begin development of a simpler, maintainable system based on simple JUnit testing and running on a Linux server. This simple development environment allowed faculty and student workers to develop method

verification software without going through the steep learning curve required for using Quiver.

We are using the textbook *Data Structures with Java* by Ford and Topp [7]. We have currently covered nine chapters and had nine programming exercises in class. Table 5 lists the exercises developed so far.

Table 5: Programming Exercises

Ch	Topic	Exercise
4	search, sort	Bubblesort with flag
5	generics	Generic bubblesort
7	advanced sorting	Nonrecurvise version of findKth method
8	collections	Make a set class from a bag class
9	array list	An in-place remove duplicates method
10	singly linked list	Inserting into a generic ordered list
11	doubly linked list	Solve the Josephus problem using simulation
14	stacks	A simple XML tag balance checker
15	queues	Find the prime factors of a number in ascending order

Students have very mixed feelings about the inking on Tablet PC activities and the “write a method” activities. The most common problem is best described as the “thrill of victory and the agony of defeat” syndrome. The programming quizzes are designed so that most students should be able to have a correct version running by the end of a 50 minute class period. Students can leave once they pass all the tests. So usually some groups are leaving after 30-35 minutes (the thrill of victory) and some students are still trying to remove logical bugs by the end of the period (the agony of defeat). To overcome this difficulty we leave the testing environment open throughout the afternoon so everyone has a chance to complete the exercise successfully. But the difficulty remains that it is usually the same groups who finish early (and leave with a bit of fanfare or boos) and the same groups that are struggling at the end of the period and need to spend additional time outside of class. One way to deal with this is to rearrange groups, but this has its own set of problems.

A similar phenomenon occurs when students are submitting inked answers to questions. Some groups are consistently submitting early while others may never submit answers and assume a more passive role of just listening. The best way we have found to handle this is to wait until almost half the class has submitted answers and then judiciously select those answers that will be discussed.

9. Conclusions

Inverting the lecture and lab in a software engineering class by using video podcasts was a pedagogical success as measured by student performance. The project component for the inverted class was more difficult yet the project grades were significantly higher. The exam grades were equivalent for both classes. The overall course grades were higher in the “inverted” class due to the higher project grades.

This gain was not achieved for free; it is evident that the students in the inverted class felt they were forced to work harder since they could not pass the class without listening to the podcasts outside of class. This additional burden was not popular with many students in their last semester of undergraduate study when they had many other concerns on their minds. However many students said this was the most realistic and practical class they had taken as undergraduates. The Spring students appreciated learning about C# and the .NET environment and reported this helped them in their job interviews. There was a true appreciation of using web services to isolate the layers in a multi-tiered system and to allow mixing of programming languages and development environments.

The success of these in-class activities is highly dependent on students coming to class prepared to solve problems. Some students complain the podcasts are boring and refuse to watch them. These students are told they can get the necessary information by reading the textbook in detail, but, sadly, very few students do this. Expecting students in the data structures class to prepare before coming to class has been a major stumbling block.

In conclusion, our results have been quite mixed. Although students perform better when using podcasts and tablet PCs, these approaches have not been popular with students. First, the students believe they are being required to do “extra” work outside of class. Some students also find the classroom activities to be threatening by exposing their work (even if submitted anonymously) to critical review. These students would prefer to sit quietly through class in a passive mode and not be involved in dynamic problem solving. Of course, there is the other side of the coin of successful problem solving by many and exhilaration when their methods pass all tests successfully. So if you elect to try this instructional approach, be prepared to deal with a wide range of reactions.

References

- [1] R. Anderson, M. Dickey, H. Perkins, Experiences with Tutored Video Instruction for Introductory Programming Courses, www.cs.washington.edu/education/TVI
- [2] B. Bruegge, A. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns, and Java*, 2nd ed., Prentice Hall, 2004
- [3] A. Campbell, R. Pargas, *Laptops in the Classroom*, Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, Reno, NV, 2003, pp. 98-102
- [4] Classroom Presenter from the University of Washington <http://www.cs.washington.edu/education/dl/presenter/>
- [5] Duke Digital Initiative website <http://www.duke.edu/ddi/>
- [6] Christopher C. Ellsworth, [James B. Fenwick Jr.](#), [Barry L. Kurtz](#): The Quiver system. [SIGCSE 2004](#): 205-209
- [7] William Ford, William Topp, *Data Structures with Java*, Prentice Hall, 2005
- [8] J. Fenwick, B. Kurtz, C. Ellsworth, X. Yuan, A. Steele, X. Jia, Inter-University Software Engineering Using Web Services, SIGCSE 2007, Covington, KY
- [9] *Gentleware website*: <http://gentleware.com/index.php>
- [10] Inter-University Software Engineering Education, DUE0341506, PIs: James Fenwick, Barry Kurtz, Xiaoping Jia, Xiaohong Yuan, Adam Steele, \$309,129, start 6/01/2004
- [11] L. Moody, G. Schmidt, *Going Wireless: The Emergence of Wireless Networks in Education*, Consortium for Computing Sciences in Colleges, 2204, pp. 151-158
- [12] P. Ractham, X. Zhang, Podcasting in academia: a new knowledge management paradigm within academic settings, Proceedings of the 2006 ACM SIGMIS CPR conference, pp 314-317
- [13] B. Simon, R. Anderson, C. Hoyer, J. Su, Preliminary Experiences with a Tablet PC Based System to Support Active Learning in Computer Science Courses, ITICSE 2004, Leeds, United Kingdom, pp. 213-217
- [14] Ubiquitous Presenter from the University of California, San Diego <http://up.ucsd.edu>