The Department of Computer Technology (CPT) at Purdue University finished its fourth offering of the Software Development for Mobile Computers course in the spring 2004 semester. This course aims to give students an opportunity to learn how to develop applications for mobile computing platforms.

The course was first offered during fall 2002 and has been offered each fall and spring semester since then. Originally listed as CPT 388M, the department gave it the more permanent course number of CPT 355 during the fall 2003 semester. Students take the course as an elective during their junior or senior year, although about five graduate students have also taken it. About 20 or 25 students finish the course each semester. Most of the students are majoring in computer technology, but students from computer science have also done well in the course.

ORIGINS

My original motivation for developing the course was simply to try to get students more interested in a career as a software developer. I started teaching programming courses for CPT in the spring of 1998, and I found that most students didn’t share my joy of writing code. Instead, they were choosing other elective courses on topics such as database management systems, network engineering and network administration, and system analysis, design, and integration.

In the spring 1999 semester, I created an enterprise application development course to teach multitier application development. Although that course is now part of our regular curriculum, it mainly interests those students who are already planning careers as professional software developers. The complexity of enterprise applications spoils the “fun” quality that I think a course needs to attract otherwise uninterested students.

I also briefly considered creating a computer game programming course. Many students expressed an interest in such a course, and I’m sure many students would enroll (at least until they found out how much work is required to develop games). But a course like this doesn’t fit with the department’s mission of giving students hands-on experience with leading edge software, hardware, and methodologies to prepare them for a career in information technology.

The idea for a mobile programming course came to me when I began seeing more and more students using Palm Pilot PDAs and cellular telephones in our hallways. I thought maybe more students would be interested in programming if I could show them how to develop applications for emerging computing platforms such as PDAs and telephones. I initially began researching software development tools for Palm OS-based PDAs. At about the same time, a professor in our department began offering a wireless networking course, and we started talking about building a mobile and wireless computing lab. Our department head and dean expressed interest in the idea and promised us lab space if we could get funding for equipment. As luck would have it, our timing was good.

While discussing our curriculum with John Spencer, our local Microsoft Uni-

QUICK FACTS

Course: CPT 355: Software Development for Mobile Computers
Unit: Department of Computer Technology
Institution: Purdue University
Instructor: Kyle Lutes
Level: Undergraduate juniors and seniors
URL: www.tech.purdue.edu/cpt/courses/cpt355
University Relations representative, John asked me if we would ever offer a course in C#, Microsoft’s new programming language. I told him the chances were slim because we had committed to use Java in all but our first programming course, which used Visual Basic. Besides, our software development courses are based on computing platforms and architectures and not on a specific programming language or product. But, I knew Microsoft had developed a PDA platform of its own, the Pocket PC, which promised to be successful. I asked John about helping us establish a course and lab for software development for mobile computing. He told me that Microsoft would soon be releasing tools that would let developers use C# to create Pocket PC applications and agreed to help fund our mobile computing project. We also secured funding for our mobile and wireless computing lab from the Eli Lilly and Company Foundation and received wireless networking gear from Intel and Kimberly-Clark. I’ve had great support from our department head, and this summer we’ll be moving the lab to a more prominent space in our building.

COURSE DETAILS

The course content falls into two categories:

- Learning about mobile computing platforms and issues involving mobile application development
- Using the software development tools to develop applications for these platforms

The technology the students use throughout the semester includes the C# programming language, Visual Studio .NET, the .NET Compact Framework (.NET CF), and Pocket PC PDAs. Our lab has dual-monitor Dell workstations, five Dell servers, a variety of wireless networking equipment, and a supply of PDA peripherals.

During the first lecture of CPT 355, my teaching assistant and I assign each student a PDA to use for the semester. I encourage the students to carry it with them at all times and use it for personal use in addition to course work. We have a variety of PDAs, including Compaq iPaq 3670s, Dell Axim X5s, Asus Razor Zayo A600s, Toshiba e740s, HP iPaq h2214s, and HP iPaq h5455s. Our business office makes the students sign a form to promise to return the equipment at the end of the semester. So far, we’ve received everything back that we’ve loaned.

TEACHING METHODS

I wish I could claim I use an innovative teaching format that keeps the students spellbound, but I would be lying if I did. Instead, I follow the model we use for many of our courses that require a lot of hands-on work. We have a one-hour lecture twice a week and one two-hour meeting in our lab once a week. During the lectures, we discuss issues involving mobile software development, and the students use the lab time to review and work on programming assignments.

We don’t require a textbook for the course. During the first two semesters we offered the course, we used beta versions of the .NET CF, and no industry books had even been published on it, let alone academic texts. Several .NET CF books are now available, and I make them available in the lab. But, for the most part, weekly readings are from periodicals and Web articles. I post a list of learning objectives for each lecture on the course Web site, along with a list of resources students can use to learn the objectives. To encourage the students to do the readings before class, I give short, unannounced quizzes about every three lectures. For the past two semesters, the students have had seven programming assignments that require them to exercise the concepts we discuss during the lectures.

For the past two semesters, the students have had seven programming assignments that require them to exercise the concepts we discuss during the lectures.

I base student letter grades on the eight programming assignments (45 percent), a semester project (22 percent), a midterm exam, (11 percent), a final exam (11 percent), and the quizzes (11 percent).

COURSE TOPICS

I use the first two weeks of the course to introduce the students to everything they’ll use during the course—the lab equipment and policies, PDAs (using, charging, installing applications, syncing with a desktop, connecting to the wireless network, and the all-important task of knowing how to reset it when a program hangs), C#, Visual Studio .NET, and the .NET CF. I also spend one lecture reviewing the object-oriented programming topics they should have learned in the prerequisite courses.

The first time I offered the course, I wasn’t quite sure about the prerequisites. During that semester, two students who’d only had two previous programming courses later dropped, citing they “felt lost” trying to complete the programming exercises. I’ve since changed the prerequisites to require three prior programming courses (introduction to programming, programming for the Internet, and object-oriented programming) and one database SQL course.

In the prerequisite courses, the students have used Visual Basic and Java. Because C#’s syntax closely resembles Java’s syntax, I really don’t have to spend much time teaching C# syntax. Instead, I’ve written a “C# Quickstart Guide” that covers what I consider to be the language’s fundamentals (data types, scope, operators, decisions, loops, methods, classes, and collections). I give this guide to the students, and we review it together after they’ve had a chance to read it. I wasn’t sure how the students would react to the “you already know
how to program so go figure out Visual Studio and C# on your own” approach, but each semester I’ve spent less and less time teaching C#, and the students have been able to pick it up just fine.

During the first few weeks, I give students their first programming assignment, which currently involves creating a simple task list application. This assignment’s purpose is to get students programming early and to get them used to Visual Studio, C#, and the PDAs. After the introductory lectures, the remaining course topics are grouped into 10 units, including the introductory lectures, wrap-up, and final project. We spend about one week per unit.

1: PDA file system, basic file I/O, and handling runtime exceptions

These topics are a bit remedial, but a PDA’s file system is a little different from a standard computer’s, and the unit gives students more practice handling exceptions—something they all too frequently ignore. We do a lab assignment in which we add simple text file persistence to the previously developed task list application.

2: Graphics programming

We barely touch on .NET CF’s graphics capabilities, but the students learn enough to develop a simple scribble pad for the next homework assignment. The techniques for letting a user make simple drawings using a stylus might seem trivial, but they prove useful later in the course. Because PDAs typically don’t have keyboards, getting information into them isn’t an easy task. Students often use the techniques they learn in this unit in the applications they develop for their semester project. For example, a couple of teams have developed instant-messaging applications in which they can scribble and send notes to each other over the wireless network.

3: Application design and user interface considerations

We discuss ways in which a PDA differs from a desktop computer. Many of these are obvious to the students (for example, the PDA’s screen size is smaller, and it has no keyboard or hard drive), but many differences aren’t so clear. For example, mobile devices are designed to be used seconds or minutes at a time and so turn on and off instantly, they are less secure because they can easily be lost or stolen, and there isn’t an easy way to print your application data (see www.informit.com/articles/article.asp?p=170200 for an entire article on these topics). Also in this unit, we study Microsoft’s Designed for Windows Mobile Software Application Handbook for Pocket PCs (see www.qualitylogic.com/alsite/certprograms/pocketpc_spec.html), and I teach the students how to use C# to make the application look and behave properly.

4: Mobile application architectures

I describe and cover the advantages and disadvantages of the stand-alone, synchronized, and always-connected client-server, the sometimes-connected client-server, thin clients, and Web applications for mobile devices. During this fourth unit, I give students their fourth programming assignment. The students must develop an application that uses the always connected client-server architecture that consumes a Web service we have running in the lab. The application must also use multiple forms and comply with the user interface considerations described in the third unit.

5: Microsoft’s SQL Server CE

SQL CE is a database engine that runs directly on the device. As with most topics, we discuss the advantages and disadvantages of using such a system on a device. Some of the benefits include referential integrity, transaction processing, and data encryption. Some of the disadvantages include deploying the engine’s dynamic linked library, the space the DLL takes on the PDA, and its price if you want to synchronize it with a server-based database engine. The homework assignment for this unit has been to modify the solution to the previous assignment to store all data on the local device rather than to save and retrieve data using the Web service.

6: Network programming using sockets

Writing your own protocol on top of TCP/IP is more complex than using a Web service, but I argue that because Web Services typically use SOAP and XML over HTTP, they’re not as efficient as a custom protocol. For a mobile application, this could be particularly important, especially for a PDA that relies on a cellular data network for data communication with a pricing plan based on how much data is transferred. Even if there wasn’t a business reason, I might still cover these topics because this is our only software development course that covers socket programming, and student comments have indicated they particularly enjoy learning the topic. For example, one student said, “My favorite part of the class was network programming and the use of threads. These are both hard to work with, but a challenge is always welcome because it makes everyone work.”
7: Threading and timers
As with the previous unit, student comments indicate they appreciate learning how to do multithreaded applications even though such applications can be tough to debug. For the final homework assignment, students modify the solution one last time to use multiple threads to continually check for a network connection and to provide feedback to the user when local data is synchronized with the server.

Wrap-up
The remaining few weeks of planned lectures include units on interoperating with unmanaged code using P/Invoke, application deployment techniques, and alternative development tools and platforms. I don’t give further homework assignments during these weeks because the students are working on their semester projects.

SEMESTER PROJECTS
Working with students on their semester projects is my favorite part of the course. I’m continually impressed with the students’ ability to come up with creative and innovative project ideas. I impose almost no constraints on what they’re allowed to do. As long their idea has something to do with mobile computing and is challenging enough, I usually allow it. Students seem to appreciate the project’s open-ended nature. One student said, “I especially liked how we had the opportunity to create our own projects. It gave us a chance to implement our own ideas and explore areas that interested us.”

During about the semester’s tenth week, the students must submit a project proposal document in which they describe what they intend to do. My teaching assistant and I suggest project ideas if someone can’t come up one. We propose enhancements if the proposed project isn’t sufficiently challenging, and we give advice on ways to reduce a project’s scope if it seems too complex. In addition to the proposal, the students must meet with us for weekly progress meetings, give a 20-minute presentation to the class during the semester’s final weeks, and complete a project post-mortem document in which they suggest the score they think they’ve earned.

Students can either work by themselves or in a team. I don’t impose a limit on the team size, but CPT’S curriculum requires a lot of group projects, so the students have learned that a large team (with its scheduling, communication, and management issues) is not necessarily a better team. Most students opt to work alone or in teams of two or three.

The projects tend to be one of two types: researching something not already covered in class or using the concepts covered in class to develop an application that solves a real problem. You can find more details of these projects on our lab’s Web site: http://mobile.tech.purdue.edu.

Research projects
I like the research-type projects because they give me a chance to learn something new, help us explore topics we might use in the course in the future, and are an excuse to buy fun new hardware to play with. Some of topics that students have covered include

- Comparisons of C# and Microsoft’s .NET CF to Java’s J2ME
- SQL server merge replication
- Application development using the Microsoft Mobile Internet Toolkit
- Signature verification
- Wireless communication using Bluetooth

Proof-of-concept applications
Students have also built applications that interface to various hardware peripherals, including

- GPS units
- Barcode scanners
- Magnetic-stripe card readers
- RFID readers
- Compact Flash Digital cameras
- Biometric fingerprint scanner on the HP iPaq h5455

Development projects
Most students choose an application development project (see the related “GoneFishing” sidebar). Some of my favorites include

- An FTP client
- A file manager replacement
- Sports-related scoring and statistics for basketball, soccer, and bowling
- A PDA-based remote control for Winamp
- Tools for instructors to communicate with students and to give quizzes using a PDA
- Several instant messaging systems
- A virtual piano and song composer
- A Smartphone interface to Purdue’s directory, news, and calendar Web sites
- A bird watcher’s field guide
- Several applications for restaurant servers
- A single-player, breakout-style arcade game

I’m continually impressed with the students’ ability to come up with creative and innovative project ideas. I impose almost no constraints on what they’re allowed to do.
My favorite example of the students’ creativity with the semester project is the GoneFishing application (see Figures A1 and A2). This project has two main programs.

The first program interfaces with a compact-flash GPS unit to draw a map of a lake. You start the program, tell it to start plotting your longitude and latitude, and drive your boat around the lake’s perimeter (or walk around buildings on campus like the students did for testing). When finished, the program optimizes the perimeter points and saves them to a map file.

The second program uses the maps created by the first. It uses the GPS unit to show the boat’s location on the lake, lets the user indicate where fish have been caught, and can show a history of where fish have been caught. You can filter caught-fish locations by lake, fish type, size, and bait used. The application also contains a database of current regulations to help you determine if your fish is legal and fish photos to help identify the type of fish caught.

“This is one of the best courses I’ve had at Purdue.”

“I have to say that I learned more in this course than any other course I’ve ever taken. I really enjoyed the topics we discussed as well as the labs. It felt good to finish a lab and know that I created an application that was actually doing something. I wouldn’t change anything about this course. It’s the best computer course I’ve taken.”

“Classes like this one are the reason why I chose CPT as my major.”

My favorite: “As a last semester senior, this was the only class I did not dread coming to. Great class.”

I plan to continually evolve the course to keep pace with industry’s use of mobile computing platforms. For example, next semester I hope to use more Smartphones in addition to PDAs. I’m also searching for an application to encourage the students to use their devices more in their daily lives. Based on a student’s suggestion, I might try having them develop an assignment calendar and grade calculator for part of their programming assignments. Recently, Intel named Purdue University as the second most wired university (www.intel.com/products/mobiletechnology/unwiredcolleges.htm), so there should be something we can have them do with all of the wireless coverage on campus.

Kyle Lutes is an associate professor in the Department of Computer Technology at Purdue University. Contact him at kdlutes@tech.purdue.edu.