Cross-Platform Mobile App Software Development in the Curriculum

Kyle Lutes
Department of Computer and Information Technology, Purdue University, West Lafayette, Indiana, USA

kdlutes@purdue.edu

Abstract

The Department of Computer and Information Technology at Purdue University in West Lafayette, Indiana has offered courses in app development for mobile devices (e.g. smartphones) beginning in 2002. Teaching mobile app development courses present many challenges to educators given that mobile device technologies are changing at a blistering pace, and there is no clear smartphone market leader. In this paper, I present my experience teaching such courses over the past 10 years, suggest a new approach that uses cross-platform development tools, and describe the pedagogy I am currently using in my CIT 355 Software Development for Mobile Devices 1 course. Finally, I include discussion on why I plan to continue using this pedagogy with cross-platform development tools.

Keywords: Smartphone, mobility, apps, software development, programming, iPhone, Android

Introduction

Since its introduction in 2007, the Apple iPhone has changed our perception of how mobile phones should look and how consumers use them. At the same time the iPhone has helped to make Apple the most valuable company in America. However, even with its runaway commercial success, the Apple iPhone is not the current smartphone market share leader. Heavy competition from Google Android, Microsoft Windows Phone, and Research in Motion Blackberry, along with fickle consumers, keep the smartphone market from being dominated by a single platform. Recent statistics from market intelligence company comScore listed the U.S. Smartphone platform market share for January 2012 as 48.6% for Google Android, 29.% for Apple iOS, 15.2% RIM Blackberry, and 4.4% Microsoft Windows Phone ("comScore Reveals," 2012). As an indicator of how quickly market share can change, the low market share for Windows Phone is predicted to increase to 20.9% by 2015 ("IDC: Android", 2012).

No clear platform leader does present a conundrum for professional software developers as each mobile device platform maker provides software development tools for their platform. Because the vendor's tools have little in common, an app that runs on multiple smartphone platforms must be written for each of those platforms. Choosing a mobile platform and toolset is especially difficult for educators who have little free time to become skilled on any one platform, let alone several. The Department of Computer and Information Technology at Purdue University in

West Lafayette, Indiana has offered courses in app development for mobile devices since Fall 2002. Each semester all students have been required to write code using the native app developer tools provided by the platform vendor. During the Fall 2011 semester I am trying a new approach of using cross-platform mobile app development tools which should allow students to develop apps for the platform of their choosing. In this paper, I discuss my experiences teaching with these tools including an overview of how apps can be built using these cross-platform technologies, our course pedagogy, the learning topics we covered, problems we encountered, and our recommendations for others considering the same approach.

Background

By design, the curriculum in the Department of Computer and Information Technology (CIT) at Purdue University in West Lafayette, Indiana is focused on the application of Information Technology (IT) rather than on theoretical computing. To this end, CIT's course learning objectives are designed so students learn through hands-on, practical experience. This characteristic differentiates our courses from those of the Computer Science (CS) or Electrical and Computer Engineering (ECE) departments, which are more theoretical based. The typical student progression through our curriculum includes a minimum background of three programming courses, including an introductory course using C#, a web application development course using HTML and C#, and an intermediate object-oriented programming course using Java (Harriger, Lutes, & Purdum, 2007).

I have been teaching mobile app development courses for CIT beginning in the Fall 2002 semester. These courses, CIT 355 Software Development for Mobile Devices 1 and CIT 425 Software Development for Mobile Devices 2 (Lutes, 2004), have proven to be popular and have been offered nearly every semester since. The typical student is a junior or senior majoring in CIT. Occasionally undergraduate and graduate students from other related disciplines such as CS, ECE, and Computer Graphics also take the courses. Typical enrollment figures are between 12 and 20 students per semester. The enrollment numbers are limited by the number of workstations and mobile devices in our department's mobile computing laboratory.

My original goal for developing these courses was simply to motivate more students to choose careers in software development. While CIT offers many elective computer programming courses, none seemed to offer the "fun factor" that we found when writing apps for mobile devices. CIT's mobile courses were first based on the Palm OS PDA platform, but were quickly changed to the Microsoft Pocket PC PDA platform as Pocket PCs were the clear market share leader in the handheld device space at the time. In addition to being the market share leader, the Pocket PC platform proved useful for pedagogical reasons. The toolset used to develop apps for this platform consisted of the C# programming language, the Visual Studio IDE, the .NET Framework class libraries, and a Windows PC – all tools our students had used in prerequisite courses.

Perhaps more importantly, the tools provided by Microsoft were tightly integrated resulting in a very productive environment for software developers. The tools worked together so well that within two weeks into the semester, our students were over any learning curve associated with using the tools. Very little effort was spent "figuring out" the quirks and bugs with the tools. Instead, student time was devoted to actual app design and development giving the course the "fun factor" we had hoped for.

From PDAs to Smartphones

As commercial mobile device technologies evolved from PDAs to smartphones, so did the courses. We switched from the Microsoft Pocket PC platform to the Microsoft Windows Mobile

Smartphone platform, which used the same developer tools used for Pocket PC PDAs. With that change, my course become slightly harder to teach as new abilities were added to the phones, new topics were introduced to the courses, and the device profiles became fragmented. Overall, teaching the course was manageable and remained popular with students. Over the next few years we found the Windows Mobile smartphone platform becoming outdated and the students expressed interest in new mobile platforms, including the Apple iPhone.

A primary reason for teaching software development courses for mobile devices is because of student interest, and my experience has shown student interest in smartphone platforms is directly linked to popularity. Heavy competition between Apple iOS, Google Android, Microsoft Windows Phone, and Research in Motion Blackberry, along with fickle consumers, keep the market from being dominated by a single smartphone platform.

This market fluctuation and heavy competition is generally seen as good because it spurs innovation and will ultimately benefit consumers. However, no clear platform leader does present a conundrum for software developers. Each mobile device platform maker provides software development tools for their platform, but unfortunately for software developers, these tools have very little in common. For example, developers for Apple iOS devices typically use the Objective-C programming language, Java is used for Google Android, and Windows Phone developers use Silverlight and C# for app development. Because the vendor's tools have little in common, an app that runs on multiple smartphone platforms must be written for each of those platforms.

Choosing a platform and toolset for professional developers is difficult enough, but the choice is especially difficulty for educators who have little free time to become skilled on any one platform, let alone several. Seeking to provide an enjoyable experience for the students similar to that of the early Pocket PC-based courses, I have reworked the courses over the past few years to use many of the popular smartphone platforms.

Smartphone App Development Options

In this paper I use the term *native app* to describe an app written using the developer tools and APIs provided by the smartphone platform vendors (e.g. Google Android, Apple iOS, Microsoft Windows Phone). Native apps typically are written using a compiled programming language (e.g. Java, Objective-C, C#), code class libraries from the vendors, a robust IDE, and UI design tools. Native apps have access to all platform specific features and are distributed through the vendor app stores.

On the other end of the spectrum is *mobile web apps*. Mobile web apps are simply web sites designed with a UI optimized for mobile devices. Mobile apps are created using standards-based web browser technologies such as HTML, CSS, and interpreted JavaScript. Mobile web apps are not distributed from vendor app stores and are instead accessed using the web browser found on the mobile device.

A *native web app* is a hybrid between a native app and a mobile web app. In this paper, I use the term native web app to describe an app that is written using HTML5, CSS, and JavaScript, but it compiled into a native app for distribution via the vendor app stores. From a user's perspective, there should be no difference between a native app and a native web app. What allows native web apps to be cross platform is that they exploit the fact that each of the major smartphone OSs contain a mobile browser that can run HTML5, CSS, and JavaScript code, and use local device storage. Open-source libraries and tools provide a shell project with the main UI being a single web browser widget that processes the HTML5, CSS, and JavaScript code packaged into the native app.

RIM Blackberry

The Spring 2009 semester of CIT 425 was taught using the RIM Blackberry smartphone platform. For pedagogical reasons, the Blackberry was a good choice. We were loaned Blackberry smartphone devices from RIM, RIM provided some instructional materials, the software development tools are free and run on Windows workstations, and the development environment is based on the Java programming language, of which CIT students have had prior experience. Although the class was well received, common student complaints included: the development environment being regarded as too similar to Windows Mobile devices, yet more cumbersome to use; and students were not interested in the Blackberry platform as few students owned Blackberry devices because of its perception of a business enterprise smartphone platform.

Apple iOS

The Fall 2010 semester of CIT 425 was taught using the Apple iOS platform that is used for iPhones, iPads, and iPods. Even though the student demand for an iPhone programming course was high, I was hesitant to teach such a course because of several perceived obstacles (Lutes, Shanklin, 2012).

Developing for the iPhone requires using Macintosh workstations. Macintosh workstations use the Macintosh Operating System (Mac OS). While many students, and a few faculty, own and use Mac computers, no prerequisite courses require any working knowledge of Mac computers. All prior development for mobile devices had been done in a Windows PC environment. In fact, there was no Macintosh computing laboratory within our department. Software development for iPhones also would require instructors and students to learn new development tools – the Xcode IDE, Interface Builder, new class libraries and SDKs, and the Objective-C programming language.

Finally, obtaining handheld device hardware would be problematic. For each semester the mobile courses have been offered, I have always provided loaner devices for each student to use. While the platform vendors provide excellent software simulators for testing and debugging, real user experience, with all the trials that this includes, can only be found using actual devices. For example, touch-based UIs, performance, sensor interaction, how an algorithm affects battery life, and data communication via phone carrier network all behave much differently on real devices compared with simulators running on a PC.

Through several sources, I obtained funding to equip our computer lab with the necessary desktop and mobile device hardware and software. The course began with a high level of excitement, but little enthusiasm remained when the semester ended. The very high learning curve associated with the iOS developer tools sapped much of the fun I originally hoped for and have seen in previous semesters. In fact, my opinion is that a minimum of two semesters is required for students to thoroughly grasp the fundamentals of iOS app development using the native app developer tools.

Windows Phone 7

Hoping to achieve the level of productivity we saw from our students with the Windows Mobile platform, we tried the newly released Windows Phone platform during the Spring 2011 semester of CIT 355. This new platform from Microsoft had been receiving good reviews and the primary app developer tools consisted of the familiar C#, Visual Studio, .NET class library, and a Windows PC. Additionally, Microsoft loaned our department unlocked phones to use in our classes.

In the end, the Windows Phone platform proved unpopular for students. Although students were skilled with C# and Visual Studio, user interfaces (UIs) for native Windows Phone apps are de-

veloped with Microsoft Silverlight and the Microsoft Expression Blend UI design tool. At the time, very little documentation could be found for Windows Phone app development, and once again a steep learning curve was present in the form of the Silverlight and Expression Blend tools. This learning curve greatly increased development time and so reduced the complexity of the apps that could be developed within the confines of a single academic semester. The learning curve, coupled with the low student interest in Windows Phone caused me to once again to re-think the smartphone platform I use in my courses.

Cross-Platform Mobile App Software Development

Because students reported an equal interest in developing apps for iOS and for Android, I decided to change the CIT 355 course once again for the Fall 2011 semester. Rather than requiring the students to use the native app developer tools supplied by the platform vendors, we are using cross-platform mobile app development tools and allowing students to develop apps for the platform of their choosing. Unlike failed cross-platform development tools of the past, such as J2ME, the current toolset actually holds promise for a workable cross-platform mobile app development environment.

Another difference is that I did not provide a smartphone device for each student. Instead, students were encouraged to use their own personal smartphones. If they did not own a suitable device, they could check out an iPod Touch, iPad, or Windows Phone device from my mobile computing lab.

Course Pedagogy

For the Fall 2011 semester, I followed the same basic strategy I have used in previous semesters with other smartphone platforms. I typically do not require a textbook for these mobile courses, and this semester was no exception. The state of the art in mobile app development is changing at a blistering pace and no current industry book, let alone a college textbook, contains current relevant information. Instead, I provide the students with a list of learning objectives for each unit and require readings from various websites found online.

As for assessments, essential app development topics are covered and each week a programming assignment is given based on that week's topic. These programming assignments are graded by myself (or sometimes a teaching assistant) by running the app and reviewing the source code. Weekly quizzes are given, as well as a midterm and final exam. Additionally, the students are required to complete a semester project of their choosing which can consist of either the development of a non-trivial mobile app, or a research paper on a topic related to mobile computing. Students can choose to work alone, or work in a team of their choosing.

The following sections describe the topics we covered in each unit, as well as my impressions of teaching that unit.

Differences compared with desktop app development

We start the course by discussing attributes unique to smartphone app development. Today's college students have likely only used computers with gigabytes of RAM, multiple overpowered CPUs, terabytes of disk space, multiple large LCD monitors, and are always connected to the Internet via a high speed network.

To give the students a different way of thinking of app development, we discuss challenges associated with mobile device development including: single-tasking OSs; limited memory; tiny screens; battery power; alternative input/output options such as touch, voice, and soft keyboards; interaction with device hardware sensors such as accelerometers, GPS, gyroscopes, and cameras;

the importance of conserving data transmission both to preserve battery power and save the consumer money on expensive data plans; and finally vendor control over the platform ecosystems.

Platform developer tool options

In this unit, the students research smartphone market share, and what tools are used for app development for the top platforms. New this semester was a discussion of using the cross-platform technologies of HTML5, CSS, and JavaScript to develop native web apps, as well as the pros and cons of native apps vs. web apps vs. native web apps. We discuss how hardware companies are continually improving processing speed in smartphones which can allow interpreted JavaScript code to perform at acceptable speeds, and the fact that both Apple and Google have used the WebKit layout engine in their mobile browsers which has made cross platform development between iOS and Android not only possible, but also practical.

We then discuss how open-source code libraries such as PhoneGap (http://www.PhoneGap.com) can be used to compile a web app into a native app, which then allows the app to take advantage of some of the hardware unique to the smartphone and additionally allows an app to be distributed through the vendor app stores.

The pros and cons of using third-party developer tools options, such as Appcelerator (http://www.Appcelerator.com) and MonoTouch (http:// http://xamarin.com/monotouch), are discussed. For pedagogical and personal bias reasons, students in my courses are not allowed to use 3rd-party tools when doing weekly homework assignments. My arguments against using these tools include: they are typically immature and buggy; 3rd-party tool companies historically lack longevity; they might have high licensing fees; they tend to be targeted at web designers rather than application coders; and most importantly, I feel I should be encouraging students to learn essential software development skills rather than how to use a vendor product. However, students can elect to further investigate 3rd-party tools such as these for their semester project assignment.

A first “Hello World” app

The purpose of this unit is to get the students familiar with the essentials necessary to develop a trivial app for their target platform. In previous semesters, this assignment consisted of installing various IDEs and simulators, understanding the need of the many project files needed, developing a simple user interface, and compiling and running the app on a simulator. Once the students can complete these steps, they know they have their development environment set up correctly.

For the Fall 2011 semester, this topic was much shorter than previous semesters because the developer tools used are the same rudimentary tools that can be used to develop trivial web sites. A text editor for typing code and web browser for testing are all that is required. Students were allowed to use their favorite text editor and develop on the desktop OS of their choosing. As all students had experience with basic web site development, the file structure of HTML and CSS documents were given a quick review. Students who planned on ultimately developing for the iPhone were instructed to test on the desktop Safari web browser and students planning for Android apps were instructed to test on Google’s Chrome browser.

Programming language overview

To add functionality to apps, the JavaScript programming language was used. While all students had done some work with JavaScript in previous semesters to add minimum functionality to web sites, I felt it necessary to teach the essentials of JavaScript from a programming language perspective. Topics covered included: variable declaration and scope; data types and type conversions; decisions; loops; functions; arrays; math functions; OOP; and interacting with DOM elements.

Debugging

No program is ever written without having some errors that have to be found and corrected, so debugging techniques were next covered. It was at this stage that frustrations began to emerge. Students were used to using modern tools and had to be shown “old-school” ways of designing, coding, and testing applications.

Modern IDEs display color-coded syntax with code “auto completion” simplifying the task of typing code. Compilers easily identify syntax errors. Students were used to using Visual Studio to perform interactive debugging by setting breakpoints and stepping through code line by line. Additionally, students were used to using the Visual Studio Forms Designer tool to easily layout a UI by dragging and dropping UI widgets onto a form.

The HTML5, CSS, and JavaScript style of development was new to most students and reckoned back to programming strategies of the 1980s. Students quickly found out that using a text editor that at least had color-coded syntax display for HTML, CSS, and JavaScript was helpful. However, UIs had to be designed by altering plain-text HTML and CSS statements, then loading in a browser to see the results. Additionally, because JavaScript is not compiled (and is case sensitive to boot), simple syntax errors were only found by running the code in a browser. If it weren’t for the fact that the Safari and Chrome web browsers have developer tools built-in that allow breakpoints to be set in JavaScript statements, variable contents to be examined, and statements to be executed line-by-line, I feel many students would never have been able to find the simple coding mistakes that would be discovered in seconds when using a more mature and robust IDE.

Persisting data

HTML5 provides two methods for data storage that are supported on both iOS and Android web apps. Local storage can be used for saving the values of a few simple string-type variables. For more complex datasets, the SQLite RDBMS engine can be used. Our students were able to use both methods satisfactorily. However, simple sequential file access is not supported as apps are not allowed access to the file system. This lack of access to simple file structures is unfortunate since storing data in an RDBMS on a smartphone requires additional coding in the form of logic to dynamically generate DDL and SQL statements, is often overkill for simple datasets, and is difficult to code and test using JavaScript.

Perhaps more importantly is the inability of an app to know when it is ending so that it can persist state information, and then load that state information the next time the app is started. When an app cannot reliably be informed it is closing, its only alternative is to save state information whenever the state information changes. This time consuming process is normally not advised on resource constrained device such as a smartphone and especially one written in a programming language being interpreted at runtime by a web browser control.

Deployment

Probably the most frustrating part of developing for iOS devices is compiling and deploying to an actual hardware device. Our development cycle consisted of first debugging and testing an app using a desktop web browser. Once it was determined the app functioned reasonably correct, the HTML, CSS, and JavaScript source code files were copied to a web server. The app was then accessed from a device’s web browser and tested again to ensure it would run on an actual device. The next step was to compile the HTML, CSS, and Javascript into a native web app using the PhoneGap tools. If the build was successful, the app could be deployed to the student’s device. However, because Apple controls which apps can be deployed to iOS devices, this process is non-trivial. A complex sequence of registering the device, registering the developer, registering the app, and registering developer’s computer with Apple must be followed. When this process

fails, it is very difficult to discover why, which results in many hours of trial and error attempts by the students.

Client/Server communication

Many mobile apps benefit by sending and receiving data to and from a server. Client/server communication can be accomplished in mobile web apps by using the XMLHttpRequest object to make simple text-based HTTP requests. While other protocols can be used for network communications, I prefer the simplicity and scalability of HTTP. Additionally, our students have had experience developing server-side code to handle HTTP requests in prerequisite courses to send dynamically generated HTML to a web browser.

To test server communication, students were required to develop a simple chat program that allowed at least two users to send text messages to each other through a server. While this programming task ultimately worked, we at first feared client/server communication would not be possible once the apps were deployed to the smartphones because the specification for the XMLHttpRequest object states that, for security reasons, cross-app HTTP requests were not allowed. Fortunately, the mobile web browsers we tested don't enforce this restriction.

Graphics and game programming

I have found that students enjoy developing games and so have included units on developing 2D casual games in my smartphone classes. We were worried that the performance and features of the JavaScript-based apps wouldn't allow this topic to be covered in a reasonable manner. However, we were pleasantly surprised to learn that it works very well. JavaScript has timer functions that allow a simple game loop to execute 30 times a second. HTML5 contains a Canvas element that comes with a reasonable set of graphical functions for drawing simple 2D shapes and text. Similarly, raster graphics can be manipulated from JavaScript allowing games to be implemented using sprite sheets.

Multimedia

We attempted to add sound to our 2D casual games but here again we found the HTML5, CSS, and JavaScript environment to be lacking. HTML5 does contain an Audio tag that can be used to play audio files such as MP3s and WAVs, however our experience showed the performance is not quite good enough to use for sound effects for an interactive game. Additionally, audio files could only be played as the result of a user action (such as touching the screen) and so could not be reliably used from within a game loop to indicate some event has occurred (such as two sprites colliding).

Interfacing with device hardware

Interfacing with device hardware is one of the unique aspects of developing apps for mobile devices. The inclusion of many sensors in the phone hardware make new categories of applications possible. Because HTML5, CSS, JavaScript are primarily used as cross-platform web app development technologies, they were not designed to directly support communicating with the sensors commonly found in smartphones.

Fortunately, the PhoneGap API does provide much of this missing functionality. PhoneGap libraries allow JavaScript code to interact with many (but not all) common sensors including: the accelerometer; compass; GPS; cameras; microphone; network connection status; and dedicated hardware buttons (e.g. volume up and down). Students are given a programming assignment to implement sensor input in a non-trivial way.

Advanced UI design and development

Developing an app using HTML5, CSS, and JavaScript and getting it to run on a smartphone is quite doable. Making that same app have the same look and feel of a native app proves quite time consuming. In fact, this unit caused us the most problems and because it was covered early in the semester, I had serious doubts about continuing the course using these technologies.

Developers use many tricks to make a native web app appear with the same UI as a native app. Most often these tricks are in the form of open source JavaScript frameworks, highly detailed CSS styling, and bitmap graphic slices used for UI object backgrounds. Based on our research into these frameworks, we can determine that mimicking the look of native apps is highly desirable for many developers, yet clearly hard to do.

I base my opinion on the sheer number of frameworks available and the many differing opinions on how to achieve what should be simple effects. In fact, we found over 80 3rd-party JavaScript code libraries and related frameworks available that profess to simplify app development. Projects such as Sencha Touch, JQuery Mobile, and NimbleKit all can be used to help a native web app look like a native app. However, we found subpar performance, sketchy reliability, and limited technical support.

These frameworks also illustrate how broken the toolset is. After all, GUI widget kits have been available for over 25 years in one form or another. If the state of the art in mobile app development requires developers to hack together a UI using many open source, free, perhaps unsupported, JavaScript code libraries and image files to do things as simple as gradient colored buttons, scrolling lists, and animated page transitions, the software development field is clearly moving in the wrong direction.

In future semesters, I plan to de-emphasize the desire to make our native web apps appear the same as native apps and will instead use simple HTML5, CSS, and JavaScript drawing techniques to present an attractive UI that may not look exactly the same as a native app, but proves more than satisfactory. I also intend to move the advanced UI design and development topics to late in the semester so that students can achieve a level of success developing usable mobile apps, before resorting to the messy matter of reviewing, testing, and coding with these 3rd-party libraries.

Conclusion

The smartphone market share continues to fluctuate providing a problem for software developers as native app development requires learning a different toolset for each smartphone platform. Educators should consider teaching mobile app development courses, but choosing a smartphone platform is an especially difficult choice for faculty who don't have the free time required to learn the developer tools for any given platform, and design and deliver a course that uses those tools. By utilizing the web development technologies of HTML5, CSS, and JavaScript along with native web app developer tools such as PhoneGap, I feel a reasonable cross-platform toolset and mobile app development course is both feasible for the instructor and fun for the students.

My experience using these cross-platform tools during the Fall 2011 semester were full of difficulties and left a lot to be desired from the software development tools when compared with the developer tools available for more mature computing platforms. Still, students were able to make more progress than what was experienced using the native app development tools for iOS, RIM Blackberry, and Microsoft Windows Phone. An added bonus is that students were allowed to develop apps for the smartphone platform of their choosing. Based on my experiences during the Fall 2011 semester, I am continuing the same pedagogy in the CIT 355 course during the Spring 2012 semester. I feel if make a few adjustments to how topics are presented, the course can once

again provide the "fun factor" I originally hoped to get from teaching mobile app development courses.

References

comScore Reveals the Top Smartphone Platform Share. (2012). Retrieved March 19, 2012, from: http://www.icharts.net/chartchannel/chart/2012/comscore-reveals-top-smartphone-platform-share

IDC: Android, Windows Phone to lead smartphone market in 2015. (2012). Retrieved March 19th, 2012 from: http://www.fiercemobilecontent.com/story/idc-android-windows-phone-lead-smartphone-market-2015/2011-06-10

Harriger, A., Lutes, K., & Purdum, J. (2007). Designing Curricula to Teach Concepts and Increase Employability. *Proceedings of American Society for Engineering Education*, 2007.

Lutes, K. (2004). Software development for mobile computers. *IEEE Pervasive Computing*, July-September 2004.

Lutes, K., & Shanklin, T. (2012). So you want to teach an IPhone programming course? *Computers in Education Journal*, *3*(1), 59-65.

Biography

Kyle Lutes is an Associate Professor for the Department of Computer & Information Technology (CIT) at Purdue University. Kyle joined the department in 1998 and is the chair of the department's software development curriculum. His teaching and scholarly interests cover a broad range of software development areas including software applications for mobile devices, data-centered application development, and software entrepreneurialism. He has authored/co-authored numerous papers and two college textbooks on various software development-related topics. Prior to his current appointment at Purdue, Kyle worked for 16 years as a software engineer and developed systems for such industries as banking, telecommunications, publishing, healthcare, athletic recruiting, retail, and pharmaceutical sales.

In addition to his teaching and research duties at Purdue, Kyle is the founder of DelMar Information Technologies, LLC. His company specializes in custom software development for mobile (smartphones and tablets), enterprise, web, client/server and desktop architectures. DelMar Information Technologies also sells many software products and services, including training classes.