AC 2007-1424: DESIGNING CURRICULA TO TEACH CONCEPTS AND INCREASE EMPLOYABILITY

Alka Harriger, Purdue University

Alka Harriger joined the faculty of the Computer and Information Technology Department (CIT) in 1982 and is currently a Professor of CIT and Assistant Department Head. The CIT Department offers baccalaureate degrees in information technology. Additionally, CIT offers numerous service courses to the university in the areas of pc literacy and programming. Professor Harriger's current interests include reducing the IT gender gap, web application development, and service learning.

Kyle Lutes, Purdue University

Kyle Lutes is an Associate Professor of Computer & Information Technology (CIT). He has authored/co-authored a numerous papers, many of which were presented at national conferences or published in trade magazines/journals as well as two college textbooks. His teaching and scholarly interests cover all areas of software development, including programming languages, mobile computing, object-oriented programming (OOP), software engineering, client/server information systems, web application development, user interface design, and rapid application development (RAD). Kyle has been writing software professionally since 1982. Prior to his current appointment at Purdue, he held various software development positions in industry.

Jack Purdum, Purdue University

Jack Purdum is currently an assistant professor in the Computer Technology department at Purdue University. He is the author of 14 programming texts and has research interests in methods in computer language education, image processing, and mobile computing. Dr. Purdum was also the CEO of a company that produced compilers, editor, assemblers, linkers, and other programming tools as well as a statistics package.

Designing curricula to teach concepts and increase employability

Abstract

The software development curriculum in the Computer and Information Technology (CIT) Department at Purdue University was designed purposefully to provide students with marketable skills upon successful completion of each course. The software development curriculum currently includes four required courses and four elective courses. The focus of these courses is on software development concepts. Students are taught how to apply course concepts using specific technologies. These technologies are selected based on their effectiveness for illustrating important course concepts as well as on industry demand. The latter is directly related to the idea of equipping students with a new, marketable skill in each course.

For example, after successful completion of the programming courses taken during the freshmen year, students will have learned HTML, introductory object-oriented programming using C#, and web application development using ASP.NET. Consequently, many freshman students have found summer internships which provide paid work experience developing static websites using HTML, Windows applications using C#, and dynamic websites using ASP.NET.

In addition to using popular technologies to illustrate course concepts, some courses provide students with practical experience building software solutions to realistic problems, often submitted by local businesses and non-profit organizations. This experience further enhances their understanding of programming concepts and helps them gain greater confidence with their newly acquired skills. In addition, class projects give them practical experience in user requirements definition and working with agents who are external to the educational process.

This paper describes CIT's software development curriculum at Purdue and shares what aspects of each course contribute to increased employability for summer internships and for part-time jobs during the school year.

Department Overview

The Department of CIT was established in 1978 under the name Computer Technology. Since that time, CIT has grown to include about 600 current majors and over 3,100 alumni. Currently, the Department offers one degree at the main campus, the Bachelor of Science (BS) in Computer and Information Technology. The BS degree provides a foundation for continued education (e.g., graduate education) and professional growth (through hands-on experiences in the classroom).

The current BS curriculum requires all students to take the same CIT courses during the first four semesters. After completing these required prerequisite courses, students can choose to take elective courses in such areas as network engineering, system administration, database management systems, systems analysis and design, project management, cyber forensics, bioinformatics, and of course software development.

The CIT Department also participates in the Master of Science (MS) in Technology degree program, offering a specialization in Information Technology. This degree provides students

with educational opportunities related to emerging and advanced information technology education to prepare future information technologists and leaders to explore, select, apply and manage information technology. The Department also offers a minor in End User Computing for majors in other schools and departments of the University. The end user computing minor emphasizes knowledge and skills for personal, work group, and enterprise productivity that can be applied to the knowledge-base of end users who are not computing professionals.

Since the inception of the CIT program until 2002 (just after the IT bubble burst), the department experienced strong placement (between 90%-100% every year) with some of the highest average starting salaries earned by university BS graduates. During the period of decline, CIT placement dipped to 83%; however, over the past two years, the program has returned to over 90% placement with impressive average starting salaries ($54,998 for 2005-06).

Purdue has several computer-related departments and degrees. For example, the CIT department to some extent competes with Purdue's Computer Science (CS) and Electrical and Computer Engineering (ECE) departments for students who are interested in a career as a software developer. However, the teaching emphasis of these departments is quite different. For example, CS offers traditional programming courses in data structures, compiler theory, and operating systems. ECE offers courses in C language programming and microprocessor design. However, the clear emphasis of the software development courses in CIT is on using programming languages to build business applications. Figure 1 shows these distinctions based on the student's role within the various layers of Information Technology. This distinct emphasis on using programming languages and tools for application development, rather than on teaching students how programming languages and microprocessors work, impacts the way the courses are taught. The authors believe teaching these skills help students get part-time jobs and internships even after the first semester.

The sequence of study in CIT recognizes that there are different types of software and differences in the way in which that software is developed. Software development is viewed here as a process of program development that embraces the entire software spectrum, from the firmware that regulates an automobile engine to a full-featured accounting system, and everything in between (i.e., compilers, linkers, assemblers, middleware, etc.). Application development, on the other hand, is a subset of software development that concentrates on design and development of end-user application software.

The distinction between these two types of development is fundamental to Purdue's computing programs. The differences between these programs are similar to the distinction between invention and innovation. Invention is the discovery of a new process or entity, whereas innovation is the commercial application of that new technology. For example, where the ECE students might design a new microprocessor, and CS students might develop a compiler for that microprocessor, the CIT student is interested in using that compiler to implement a specific business application that runs on the new microprocessor. The nuance of that distinction has significant implications on the courses each major pursues.

Figure 1: Layers of Information Technology

Roles by Major	*End User*
CIT students design, build and integrate	Presentation / User Interface (Windows, Web, Mobile Devices)
CIT students design, build and integrate	Business Application Components and Services
CIT students use CS students design and build	Development Tools (programming languages, IDEs, compilers, RDBMSs, SQL, HTML, XML, Web Services, UML, etc.)
CIT students use CS & ECE students design and build	System Software (operating systems, device drivers, network protocols, etc.)
CIT students use ECE students design	Computer Hardware

CIT: Computer and Information Technology
CS: Computer Science
ECE: Electrical and Computer Engineering

The CIT major must be able to communicate with the end user in order to design an application that meets some specific need as described by that end user. As such, CIT majors are required to take courses in English, speech, accounting, economics, and other non-programming courses. These courses enable the CIT major to communicate effectively with the (often business) end user using a lexicon common to both. Additional courses in management and leadership augment the learning experience by providing additional team-building and leadership skills. Indeed, all of the required programming courses for a CIT major have a team project as part of the curriculum to reinforce these skills. Exposure to these skill sets is also consistent with the career path of most professional software developers. That is, many CIT majors begin their careers as programmers but migrate to middle and upper management over time. The job of educators is to provide a foundation that makes that transition viable.

CIT Curriculum

Based on advice from the CIT department's industrial advisory board (IAB), all students are required to take the same courses during their first four semesters. The relevant courses include courses in computer programming, web application development, systems analysis and design, computer architectures, and database management systems (DBMSs). The members of the IAB

feel that any student who plans a career in IT should understand the essentials of all these areas. During the fifth through eight semesters, students can get more depth in any of these areas by enrolling in the many elective courses offered.

Considerations for the Software Development Courses

The purpose of this smorgasbord of course experiences is to give the student an early, yet fairly complete, overview of the curriculum areas offered by CIT. In addition, at the end of each course, students acquire a new, immediately-marketable skill. This exposure not only increases their employability for summer internships or full-time jobs, but also should help them make a more informed decision regarding the specialty area, or *track*, which they wish to pursue as upper-class students.

It is also important to emphasize that the CIT curriculum decouples the concepts of a given course from the language or tools used in the course. The language used in a course is a vehicle for teaching concepts, not an end in itself. Unlike some competing institutions that feature the language (or other tool vehicle) prominently in the course title (e.g., *Visual Basic Programming*, *Modeling Using Visio*, etc.), CIT's courses are designed around concepts rather than vehicles. (See below.) Course titles are decoupled from their tools not only because of CIT's program philosophy, but also because it allows CIT to be more agile in responding to industry needs.

Spinellis provides guidelines for choosing a programming language; however, these guidelines are appropriate for a developer about to undertake a software project.[1] When selecting the best tool to support programming instruction, the academic environment itself influences the selection. For example, in the web programming course at Purdue, a change from JSP to ASP.NET was necessitated by the fact that the servers to support instruction could not be adequately secured. Another issue that influenced the decision to change was the inability of clients for the course's service learning project to make a swift and affordable transition from the school servers to a hosting service that supported the technology used. TIOBE's index on the popularity of various programming languages can help businesses and schools make strategic decisions regarding which language(s) to use.[2] Scriptol.org provides a comparison table that lists popular programming languages, each with reasons why one should consider using it. [3]

Regardless of the actual language or tool selected, CIT's methodology for teaching concepts is also different than that found at many other institutions. For example, as much as possible, programming assignments are geared towards solving business problems. If nothing else, these assignments help familiarize CIT students with the jargon found in the business world. As stated earlier, most of these courses require an end-of-semester team programming project designed to solve real-world problems. These projects not only tie the course concepts together “in one place”, but also enhance the communication skills and team dynamics of CIT students. In addition, students use the technologies in the same way that they are used by industry. This equips the students with relevant work experience in the classroom and allows them to be productive on the job immediately. Again, the course projects use the latest technologies available, but using those technologies is simply a means to an end. As a result, students have gained experiences that have real and immediate value in the marketplace.

Another important consideration for the software development courses is to instill in the students that software development is not only rewarding, but also fun. Furthermore, through the course activities completed by students, it becomes apparent to most that they have the ability to become professional software developers. It also helps to share newsworthy information such as a recent salary forecast that stated that IT salaries will be on the rise in 2007 and software developers will be the ones with the biggest gains.[4]

If the introductory courses are taught correctly, the student should have no difficulties being successful in the advanced elective courses. Elective courses are tailored to accommodate a variety of platforms and topics (e.g., e-commerce application development, programming for mobile computers, legacy system integration, etc.).

Relevant Courses Required for all Students

When designing the CIT curriculum, the faculty wanted to ensure that the courses were organized to build on the concepts taught in earlier courses. Another goal was to expose students to different languages/technologies, so they could discern their similarities and differences as well as enhance their technical skill set to increase their marketability to employers. Each of these courses is described in greater detail in the following sections. Figure 2 at the end of the course discussion depicts how the required and elective courses fit in the CIT program.

141: Internet Foundations, Technologies, and Development

This course explores the history, architecture and development of the World Wide Web. Current tagging and scripting languages are covered in a tool independent environment. Topics also include authoring tools, design, graphic and multimedia formats, and commerce, implementation and security issues. Students use the HTML language and its more stringent counterpart, XHTML, to develop materials for the Web. Assignments require students to construct static web pages that include tables, frames, forms, audio, video and graphics components.

Students in the course acquire a basic understanding of the Web architecture and markup languages (HTML, DHTML, XHTML, XML, XSL, and CSS). They also learn minimal client-side scripting using JavaScript. After successful completion of C&IT 141, a number of students have gained internships developing static websites for small businesses.

155: Introduction to Object-Oriented Programming

This course introduces fundamental computer programming concepts. Topics include: problem solving and algorithm development, programming standards, variables, data types, operators, decisions, repetitive structures, modularity, arrays, sequential files, user interface construction, software testing and debugging, all within an object-oriented programming framework. The concepts and skills learned in this course are transferable to a wide variety of contemporary programming languages and software development tools. The C# programming language is currently used to teach concepts, although Delphi, Visual Basic, and Visual Basic .NET have been used in the past and Java certainly could be used. A recent Course Technology conference had a presentation on ten reasons why C# should be used to teach introductory programming.[4]

Each new set of skills is demonstrated through a weekly programming assignment in C#. During the second half of the semester, students work in teams to build a mid-scale, GUI, desktop, single-user software application based on the needs of a specific client. Each team presents the final project, sharing how course concepts were applied to address unique client needs. The concepts and skills learned in this course are transferable to a wide variety of contemporary programming languages and software development tools.

180: Introduction to Systems Development

This course introduces information systems development. Topics include exploration of the various types of information systems, system development, database management systems, and problem solving. Students are expected to read/create UML, ERD, and data flow diagrams to model information system objects, data, processes, and logic. Labs emphasize modeling and SQL/QBE querying to prepare students for later systems, programming, and database classes. As much as possible, students are asked to document user requirements, from which they design, construct, and test a personal computer information system. To supplement those goals, students currently use Visio and Access as part of the tool set to design and build a small desktop database application.

255: Programming for the Internet

This course is a fast-paced study of advanced Internet application development that builds on the knowledge gained in both prerequisite courses. Throughout the course, students use the various Internet technologies taught in C&IT 141. The course begins with simple Web programming that leverages the pre-built controls offered with .NET and moves to the development of complex, dynamic, database-driven sites.

ASP.NET with C# is currently used in the course. In the past, classic ASP with VBScript, Java Server Pages (JSP), and ASP.NET with VB.NET have been used. PHP and Perl have also been considered.

All essential course concepts are presented in lectures, and small group meetings in computer labs are used to give students the opportunity to practice newly-learned concepts in a supervised setting. All students are expected to use web resources to expand their knowledge beyond the classroom and share some of these newly discovered ideas with the class. An important course activity that prepares students for real jobs is a service-learning semester project in which they work as a member of a team to develop a high-quality, database-driven website for a local client. Student feedback about the team project confirms that the experience not only solidifies their knowledge of course concepts but also increases their self-confidence as a future IT professional.

272: Database Fundamentals

This course is designed to introduce students to good relational database design, including modeling and normalization. The students learn how to design entity relationship models for business problem domains and verify the design through third-normal form. After completing the

class, students should be familiar with a database data definition language (DDL), data manipulation language (DML), data control language (DCL), and the components of Structured Query Language (SQL). The student should be able to manipulate tables, indexes, and the associated data within a database using SQL statements and understand the concepts associated with database transactions, commits, and rollbacks. The student should also have a basic grasp of the role DBMSs play in client-server architectures. Oracle is currently used in this course because many employers who hire CIT students have a need for Oracle skills. However, any modern DBMS, such as IBM DB2, Microsoft SQL Server, and even MySQL could be used.

295: Object Oriented Programming

The introductory programming courses (C&IT 155 and C&IT 255) are grounded in Object Oriented Programming (OOP) concepts and those courses are required for this course. C&IT 295, therefore, extends to knowledge base formed by those two courses and places greater emphasis on the areas of inheritance, polymorphism, interfaces, class and API design, and other OOP topics. The primary vehicle is currently Java, but C++ has been used in the past and C# has been considered. (Generally, the language used in this course is different from the languages used in the two introductory courses simply to provide exposure to a different programming language.) The ante in this course is increased and students are expected to write more complex, server-side applications. Over time, this course has evolved to a greater emphasis on integrating the web programming experiences from C&IT 255 and the database knowledge gained from C&IT 272 into a fairly robust web-based application. The course team project frequently results from business problems submitted from the local business community. This synergy between the university and the local community is a win-win situation for both parties plus the students gain valuable real-world experience. It is not uncommon for students to gain internships or even fulltime employment with firms who participate in the program.

Relevant Elective Software Development Courses

Figure 2 shows where the elective courses fit within the overall curriculum. Because the faculty's goal is to focus on concepts that cross platforms, even these courses are designed to be technology-agnostic. Each of the elective courses is described in greater detail in the following sections.

355: Software Development for Mobile Computers

This is an advanced programming course that teaches students the skills necessary to develop applications for mobile computing devices (e.g. PDAs and smart phones). This course gives students hands-on experience with the technologies, tools and techniques used to develop mobile software solutions for business. Topics include introduction to mobile devices, exception handling, data persistence, graphics programming, UI design and development, mobile application architectures, micro database management systems, network connectivity using Bluetooth and WiFi, threading, and application deployment.

This course has used Windows mobile-based devices along with C# and the .NET Compact Framework (.NET CF). However, the instructors of this course have recently begun

investigating the use of J2ME (Java 2 Platform, Micro Edition) for possible use in future semester. When first offered in 2002, this was the first course in the curriculum that used C# and .NET. Students completing this course were able to find internships because of their experience with these emerging technologies.

405: Software Development Methodologies

This course explores methodologies and practices commonly used in contemporary software development projects. This course is similar to a typical Software Engineering course except that, because students have already taken several analysis and design courses, those topics are not covered. Instead, more time is spent on topics such as development methodologies, programming standards, code ownership and accountability, source code management and version control, productivity and quality metrics, software testing, intellectual property, and software process maturity models. In general, this course covers activities software developers do besides develop code. These topics become even more relevant as the students progress from programmers to assume greater middle management responsibilities.

450: Enterprise Application Development

This course explores advanced application development techniques in a large enterprise-wide setting. Topics include pros and cons of object oriented programming, database connectivity options, multi-user considerations, concurrency problems and resolutions, component development and reuse, multi-tier applications, distributed object technologies, web services, data marshalling, transaction processing, and application installation and deployment issues.

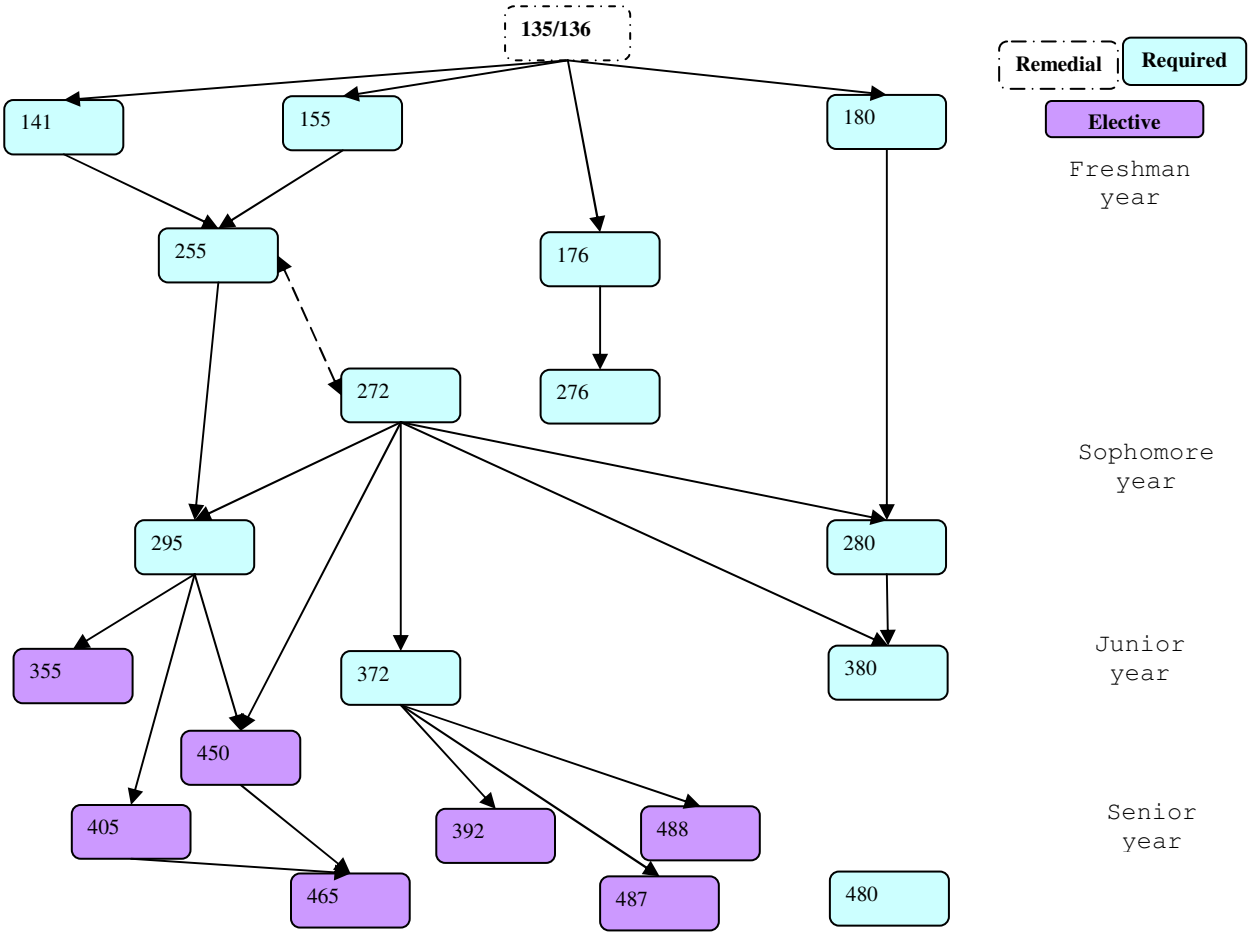
This course is currently taught using the Microsoft .NET enterprise environment, but has also been taught using Visual Basic 6, and the Java enterprise environment. Although this course is most often taken by seniors who are not looking for internships, alumni who have taken this course have reported to faculty that learning the topics from this course have given them a substantial head start on the job because other recent college graduates from other institutions have never learned this material.

465: Senior Software Development Project

This capstone course integrates the software development technologies and techniques taught in prior courses. The purpose of this course is to give students experience working on a large project team, to develop a real system of real value and quality. In most other courses, students have worked in a small team of two or three students over several weeks to complete a semester project. In this course, however, the goal is to have the students work on a project that is as close to a real development project as possible given the constraints of the academic calendar. Students work in teams as large as 12 members and spend the entire semester working on a single project. Each student volunteers for a role, or roles, on the team. Roles include project manager, systems administration, database administration (DBA), user liaison, requirements analysis, developer, tester, technical writer, and deployment specialist. Technology used in this course is dependent on the project but typically consists of either the Microsoft Enterprise Environment of .NET, ASP.NET, and C#, or the Java Enterprise Environment. Additional

support systems such as collaboration tools, bug tracking, source code control, and automated testing tools are used.

Figure 2: Critical Path of the CIT IT Curriculum



Conclusion

In the words of reviewer two (email communication of abstract acceptance, November 9, 2006), "Balancing theoretical instruction with practical application, especially in software engineering, can be challenging." Nonetheless, finding the right balance can result in a valuable benefit to the student – increased competitiveness for internships and permanent jobs. This paper described how the CIT program at Purdue University has accomplished that goal through selecting programming languages and technologies that both augment instruction of course concepts and increase marketability of job skills. This approach has directly impacted CIT's placement rates, which will be shared at the presentation. Finally, anecdotal evidence, including unsolicited comments from students, prospective employers, and the department's Industrial Advisory Board, supports the use of this approach. For example, one employer that actively and consistently recruits CIT students specifically for their technical skills shared the following comments (personal communication, January 22, 2007):

"We value the CIT students we hire each semester. They are skilled at both dealing with the high level business needs as well as the highly technical side. They are also well rounded in all the cutting edge technologies that industries value so much today.

Currently, our CIT students work to develop and maintain (our website). Their duties include web development (in ASP.NET, C#, XML, XSLT), data management (SQL), application development (C#), and system's analysis. They transition across these multiple roles with relative fluidity making them assets to our department."

If a computing program can effectively teach the same concepts using a variety of tools, why not choose the technology and tools that bolster the students' marketability? CIT has found this approach not only increases the interest of employers in CIT graduates, but it also increases the confidence of the graduates in their technical skills and reduces, in some cases, the amount of training CIT graduates must complete when first hired by an employer.

Bibliography

[1] Spinellis, D. (July/August 2006). Choosing a programming language. *IEEE Software*, 23(4): 62–63.

[2] TIPBE Software. (January 2007). *TIOBE programming community index for January 2007*. Retrieved January 10, 2007, from the TIOBE Software website: http://www.tiobe.com/index.htm?tiobe_index.

[3] Scriptol.org. (n.d.). Popular programming languages. Retrieved January 10, 2007, from website: http://www.scriptol.org/choose.php.

[4] Lutes, K., Purdum, J., Harriger, A. (2005). Ten Reasons to Use C# to Teach Introductory Computer Programming. The Conference 2005 Orlando, FL: Course Technology.