Systems Capstone Projects: On which platforms should we have our students perform?

Jay Hettiarachchy
Computer Information Systems Department
Ferris State University
Big Rapids, MI 49307
hettiarj@ferris.edu

Abstract

Educators training students in Computer Information Systems discipline face many challenges today in choosing platforms – computer hardware, operating systems and software. Designing physical layout of systems projects has become one of the most difficult challenges. In this paper, the author traces his learning experience with students taking the senior capstone course from him within the past 20 years. Within this period many changes took place in the hardware and software industry. While Client/Server computing has become pervasive in information industry, object oriented design and object oriented programming have made remarkable changes in the way we design and implement systems. Web presence and gathering and manipulating data dynamically has become a necessity in today's work place. In concluding remarks, the author discusses the implications of these changes on curricular development and presents some strategies that promote learner-centered best practices in the field of information technology.

Introduction:

The term "platform" as it is used in association with computers and computer application systems may mean different things to different people. To some a "platform" may mean the hardware on which a given application program runs. More specifically, this is a "hardware platform" such as x86, SPARC, PowerPC or Alpha. To others, a "platform" may be an operating system on a given hardware platform. For example, Linux, the various flavors of UNIX, various versions of Microsoft Windows, Mac OS, AIX, Solaris etc. But what about other large programs such as Database Management Systems (DBMS), program languages etc. which run on specific and proprietary hardware and software platforms? Examples are, DB2, MSSQL, for DBMSs, and Java, VB.NET, ASP.NET for programming languages. Furthermore, the diversity of platforms based on hardware, operating systems, database management systems, and programming languages have brought about a new generation of "cross platform" applications that are "platform-independent." Such applications are capable of performing their tasks on many different hardware platforms and operating systems. The Internet and the Web are among the greatest success stories of platform independence. Some programming and scripting languages such as C, Java, Perl, PHP and Python have significantly contributed towards platform independence. Portability across multiple platforms seems to be the main goal of open source software developers.

This paper is an attempt to address the challenges faced by educators in preparing students to face the realities of the work world consisting of such multitude of different "platforms."

Platforms and Capstones:

The capstone course is an opportunity for students to demonstrate that they have achieved the goals for learning established by their educational institution and major department. Ideally, the course should foster interdisciplinary partnerships among university departments and help cultivate industry alliances and cooperation. Capstone courses in Computer Science and Computer Information Systems disciplines have the same broad objectives. The rapid growth in the computer and information industry has however presented significant challenges in achieving these goals. A brief description of the challenges involved in this field within the past 20 years is given below.

In the late 80 and early 90s capstone computer applications systems in most colleges were primarily developed using either mainframes or mid-range computers. The author remembers the time when he trained students on IBM mainframes and mid-range hardware with IBM operating systems, IBM proprietary database management systems and programming languages, Digital Equipment Corporation's (DEC) hardware and operating systems, and many other different hardware and software such as UNIVAC, Perkin Elmer and Data General hardware and operating systems and programming languages. Most colleges during that time could only afford a single hardware and software platform -- one mainframe computer or mid-range computer when it came to supporting academic computing. There was a strict separation between academic computing and administrative computing due to security reasons, with the result that industry-standard DBMSs like IBM DB2, Oracle,

Informix, and Adabas were not available in colleges for student use. The result was that there were very limited industry-standard hardware and software resources available for developing senior capstone projects. Most senior capstone projects were developed using procedural programming languages such as BASIC, COBOL and RPG II, III languages, mostly using conventional "flat files." The availability of an IBM AS/400 with RPG 400 with a DBMS similar to IBM DB2 database management system at a college for senior capstone course was truly a luxury in the early 90s.

The computing landscape started to change in the 90s with the growth and development of micro-computers, micro-computer operating systems, micro-computer network management systems, client/server systems, and the Internet. It was also a time when micro-computer hardware platforms changed rapidly and radically. Micro-processor manufacturing industry together with the operating systems manufacturing industry took "quantum leaps" during the 90s. While the popular PC Disk Operating Systems (DOS) went through several incarnations, several versions of Windows operating system, OS2, Windows NT, and Local area networks such as Novell Network were invented during this time. Micro-computer versions of almost all popular programming languages, database management systems such as dBase, Rbase, Sybase, Access, MS SQL, and MySql gained popularity both in educational institutions and industry. The popularity of the Internet and the Web gained in leaps and bounds during the same time transforming stand-alone micro-computers to Client/Servers that enabled micro-computers to do many of the tasks that were previously done primarily by the mainframes and mini computers. In reality, Client/Server computing technology "crept into" mainframe and mini computer market of the previous years. The author had many students do capstone projects using dBase, Rbase5000 DBMSs on DOS machines in the 80s and early 90s. However, as in 1994/5, one CIS capstone project the author guided consisted of creating a Web Server for the college by the graduating seniors. This project effort perhaps coincided with the change in focus of applications development in the late 1990s.

In the wake of all these technological development – hardware, operating systems, DBMSs, object oriented programming languages, and Client/Server computing -- there was a significant shift in demand from conventional hardware, and operating systems to Client/Server computing, both in the business sector and in the academic world. Naturally, students in the CIS programs wanted to learn the technological skills that are in high demand in the work place. Consequently, the gap between the old and traditional way of using traditional conventional hardware, operating systems, and database management systems for applications development, and the new way of using Client/Server systems with object oriented programming languages and Internet-based application systems development widened significantly.

Capstones and the World of Objects:

When it comes to software platforms, the buzzwords in the computer industry are: simple, object-oriented, distributed, robust, secure, architecture-neutral, portable, high-performance, multithreaded, and dynamic. Whereas all programming languages used in the previous generation were "procedural programming languages", programming languages starting with

Simula through C++, are centered on creating objects, manipulating objects and making objects work together by re-using them. Java, Visual Basic. NET, ASP.NET are popular object-oriented languages that belong to this category.

Object-oriented programming models the real world in terms of objects. Programmers define not only the data types and data structures, but also the types of operations that can be applied to the data structure thereby making data structure an object that include both data as well as the associated operations. In manipulating objects, programmers can create relationships between one object and another. Since objects can *inherit* characteristics from other objects, a programmer can simply create a new object that inherits many of its features from existing objects. This makes object-oriented programs easier to modify.

However, switching from procedural programming languages to object-oriented programming is not always an easy transition. Object-oriented programming involves learning to use object-oriented systems design methods as well. Procedural programmers need to not only unlearn most conventional programming practices, they have to learn to adopt new methods of designing applications systems using object-oriented design methods. Such a transition in colleges seems to create the following difficulties. 1) Operational costs involved in switching to new platforms 2) Difficulties associated with selecting a suitable platform that would satisfy all contending parties 3) Having to maintain the existing platforms to facilitate the existing curricula using traditional hardware, operating systems, programming languages like COBOL, RPG, Pascal (in some colleges), and not having enough resources, both material and human to meet the needs of the transition satisfactorily 4) Keeping up with rapid changes in the industry.

A parallel development that has significantly impacted the course of development of capstone courses in computing discipline is that of the Internet. Although it is generally assumed that developing large scale systems projects as classroom projects is an almost impossible task within the time limitations of semesters, currently there seem to be a growing demand and student interest in developing applications systems that are Internet based and available from anywhere in the world.. The recent growth of e-commerce industry and the high demand for systems analysts and programmers experienced in Internet-based application development greatly contributes to this demand. Publishing and dynamically manipulating data and information on the World Wide Web has become one of the most important needs of the application systems development industry. The ability for users to interact with a business, collect and process mission-critical information in real time, and provide new levels of user support have become increasingly important aspect in applications development.

On which platforms then should we have our students perform?

Obviously, the demand for developing capstone courses on traditional platforms using procedural programming languages seem to be declining. At the same time, the requirements for developing systems using programming languages and tools that are greatly demanded in the work place and “easer to learn” seems to be on the rise. Further, programming languages

that seamlessly interface with almost all of the popular DBMSs and the World Wide Web seem to attract applications developers as well as students. Today, not only are there a large number of contending platforms available, but also they seem to have a shorter productive life cycle with each claiming to provide "newly improved and enhanced versions." The Web will attest to such claims made by almost all parties that strive to set industry standards. With each new version these products have added more functions to their newer versions, but at the same time they have become increasingly complex as well. The author's experience in working with the changing technology has been that there is a considerable amount of unlearning as well as new learning to do in transitioning from one version to another. The transition from Visual Basic version 6 to VB.NET and ASP.NET is a good example in this respect.

Within the past three years the author encouraged his graduating class of CIS students to develop and implement computer applications on several different platforms. The rationale for such experimentation and the associated challenges form the discussion of the rest of this paper.

The availability of a multitude of platforms that are competing for dominance in the industry as well as the market place for developing applications was one reason for experimenting on platforms other than mainframes and mini computers. Another more important reason was to give the graduating students an opportunity to gain hands-on experience in developing applications systems using popular Client/Server-based software. A third reason was the short period of time – one semester – available for developing a fully functional and Web-enabled application system that manipulates a back-end database such as MSSQL or MS ACCESS. One platform that fits all these criteria is ColdFusion Server working in tandem with Internet Information Server (IIS) and MS-SQL Server running on a Windows NT Server. Obviously, such a "platform" consists of a number of hardware, operating systems, database management systems, a Web Server including ColdFusion programming environment working in harmony.

ColdFusion was developed by the Allaire Corporation to be a simple to use yet powerful alternative to Perl and other CGI technologies. Allaire Corporation was recently acquired by Macromedia the same company that developed Flash, Shockwave, Dreamweaver and many other high impact web authoring tools. ColdFusion is an application that runs on a web server. It runs on Linux, Solaris, and Windows Servers. The ColdFusion Web Application Server works with the HTTP server to process requests for web pages. Whenever a ColdFusion page is requested, the ColdFusion Application Server executes the script or program the page contains. ColdFusion interacts with Database Management Systems such as Sybase, Oracle, MySQL, SQL, or Access by using standard SQL (Structured Query Language), thereby enabling web applications to retrieve, store, format, and present information dynamically. Unlike JavaScript, and Java Applets, which run on the "client", or "web browser", ColdFusion runs on the Web Server. This means that ColdFusion scripts (tags) will run the same way on every browser. However, ColdFusion also comes with a price tag that makes it harder for colleges to keep up with site licensing and upgrading for newer versions.

A second platform used for implementing applications was a Linux Server running PHP and MySQL database. The following excerpt provides a good summery of PHP capabilities:

PHP is an established server-side scripting language for creating dynamic Web pages. As a language that has been designed expressly for the Web, it brings many features that commercial entities are looking for: Exceptionally short learning curve, quick development time, and very high performance. This is essential for companies who are faced with scarce skilled programming resources and ever-tighter time to market deadlines. In addition, PHP supports all major platforms (UNIX, Windows and even mainframes), and features native support for most popular databases. All these factors make it a very good choice for Web development. [Those who] work with PHP have reported being able to hire non-programmers and have them producing usable code within days. Programmers familiar with languages such as C, C++ or Java frequently find that they can begin programming in PHP within a few hours. The fact that PHP was designed specifically for Web development gives it an edge as a development tool, as Intranet Design Magazine explains: "PHP was built with the needs of Web developers in mind... Unlike other cumbersome, overhead-laden approaches, PHP is lightweight and focused on the Web.[1]

One of the advantages of using PHP is that it comes free. Students who have a good foundation in procedural programming and object-oriented programming are able to pick up PHP scripting easily.

The author's third choice was ASP.NET platform on Windows Operating System and the Internet Information Server.

ASP.NET is designated by some experts as a technology rather than a programming language. The technology is accessible via a programming language. Therefore one can create Web pages using Visual Basic.NET and using ASP.NET to drive it.[2]

However, ASP.NET has a steeper learning curve - the operating systems, databases, programming languages, and the IIS Web Server needed to set up the "platform" for running an ASP.NET work environment that comes with a price tag and network expertise on the part of the server administrator. Although students tend to think that learning ASP.NET will have its payback time in the job market, most students today tend to shy away from wanting to develop applications systems on "platforms" that demand a good deal of programming.

Conclusions:

Advancements in computer hardware, operating systems, database management systems, and object-oriented programming languages within the last two decades have brought about many challenges of a practical nature in the work world as well as in the educational world. Responding to the demand for educating and training computer professionals able to provide ubiquitous, secure and high-performing computing applications within a short period of time with minimum conventional programming expertise seems to be one of the greatest challenges for educators today. Keeping up with technological advancements of all the latest

programming languages and satisfying student demands for hands on experience with the available hardware and software resources in educational institutions is a challenge by itself. Maintaining several hardware and software platforms for academic use can only be successfully achieved by involving students in setting up and maintaining such platforms. Students who have had a solid foundation in procedural programming and object-oriented programming languages continue to demonstrate that they could easily adapt themselves to building applications systems using the latest software platforms and tools. This of course should not surprise anyone who had a solid educational foundation in computing.

References:

1. http://www.zend.com/zend/art/php-over-java.php

2. Beginning ASP.NET with Visual Basic.NET, Rob Birdwell et al, Wrox Press Ltd, 2002.