The Complexities of Effectively Teaching Client-Server System Development

Richard M. Stillman
rstillma@nova.edu
School of Computer and Information Sciences
Nova Southeastern University
Fort Lauderdale, FL 33314, USA

Alan R. Peslak
arp14@psu.edu
Information Sciences and Technology
Penn State University
Dunmore, PA 18512, USA

Abstract

Developing real-world enterprise systems requires the integration of a number of complex technologies. The 2006 Model Curriculum for Graduate Degree Programs in Information Systems recognizes this need and recommends an integrated capstone experience directed toward helping the graduate student amalgamate the broad knowledge base required for successful development of enterprise systems. Conveying a practical knowledge of client-server systems development to computer and information systems graduate students can be challenging. In this paper, we summarize our approach to teaching and evaluation in masters' level client-server computing and software engineering courses and our methods for overcoming obstacles, and we report on some novel solutions submitted by our students to the course's final project. The grades on the final project correlated with performance on the more traditional assignments but the correlation was low, representing only 5% of the explained variance. We propose that traditional text and narrative assignments represent a different skill set from client-server system development; therefore, the incorporation of both activities is beneficial to software skill development.

Keywords: Higher education, client-server systems, multitier systems, database systems, software engineering, information systems, active learning environment, model curriculum, capstone experience, enterprise systems.

1. INTRODUCTION

Online retail sales grew from $85 billion in 2004 to $125 billion in 2007 ("U.S. Census Bureau News," 2008). The value of web-enabled business resides in the ability to combine and exploit information technology (Barua, Konana, & Whinstonm, 2005). The higher order functionality of web-enabled business not only improves operational performance, but can also positively impact customer relationships and marketing. It is the client-server system that provides this functionality.

Developing a real-world client-server system requires the integration of a number of complex technologies. The 2006 Model Curriculum for Graduate Degree Programs in Information Systems (Gorgone, Gray, Stohr, Valacich, & Wigand, 2006) recognizes this need and recommends an integrated capstone experience directed toward helping the graduate student amalgamate the broad knowledge base required for successful development of enterprise systems.

Conveying a practical knowledge of client-server systems' development to computer and information systems graduate students can be challenging (Chung & McLane, 2002; Stiller & LeBlanc, 2006; Tikekar & Wilson, 2001; Yue & Ding, 2004). In this paper, we summarize our approach to master's level client-server computing courses, and we report on some novel solutions submitted by our students to the courses' capstone projects.

Many pedagogical researchers have found the project approach to be a better approach to learning software design and development than traditional textbook, lecture, and small problem based approaches. For example, Kovacs and Baugh (2007) found "the project approach to be effective in increasing student motivation and improving student problem-solving, providing students with an integrated learning situation, and addressing different learning styles." Henry and LaFrance (2006) emphasized the importance of the kind of active learning that can only be achieved by engaging students in relevant projects. Myers (2007) observed that an effective software development assignment should be a substantial venture in a consequential domain, not a "toy" project. Other researchers who have had similar results include Albanese and Mitchell (1993), Hutchings and Wutzdorff (1988), and Stillman and Peslak (2007).

Conveying a working knowledge of client/server computing is vital at all levels of information systems' education (Lefkovitz, 2000; Raoufi, Spoa, & Wiggins, 2001). Multi-tiered systems have, of course, become indispensable throughout the corporate domain, and yet development of efficient, robust, and secure systems remains a complex endeavor (Ramsin & Paige, 2008). In fact, with the trend toward increasing horizontal integration (Foster & Tuecke, 2005), a working knowledge of client-server system development is becoming even more vital (Lawler, Raggad, & Howell-Barber, 2007; Peslak, 2006).

The educator must walk a delicate line between the need to impart this kind of practical knowledge and the risks of unrealistic requirements and information overload.

2. METHODS

The goals of our client-server and software engineering computing courses are to convey both an academic understanding and a reasonable level of practical proficiency in the concepts and principles of client-server architecture, protocols, networks, and distributed computing; distributed application design and implementation; inter-process communication; the role of the GUI and front-end development tools, middleware; multi-tier architectures; distributed objects; database interaction; relationships between client-server computing and business processes; migration from legacy systems; and meeting customer requirements.

We attempt to meet these goals through a multi-pronged approach that includes literature review, research exercises, analytical and programming assignments, and capstone projects. One of the most important ways to understand client-server computing is actually to develop a comprehensive client-server application. Our Client-Server computing graduate level course includes specific instruction on small applications but culminates in an original project that asks the student to combine this prior knowledge into a comprehensive client-server application.

The capstone project in the client-server course requires the student to develop a comprehensive proposed client server solution for an e-commerce business. The proposal must compare and recommend software, hardware, systems, specific vendors, architecture, a distributed paradigm, middleware, and database. The course leads up to the final project by having the student navigate through small assignments. Each of these assignments exercises the skills needed for one of the technologies that the student will have to combine and coordinate in order to successfully develop the client-server systems for the final project.

The capstone project in our software engineering course is a three-tier project of the student's choosing based on their background and expertise. This project is the culmination of textbook exercises in software engineering as well as targeted object-oriented development exercises. Thus, designing these capstone projects required addressing each aspect of enterprise system development: the basic three layers of client-server technology and a number of vital cross-cutting concerns.

In general, a client-server system operates on three conceptual levels: the presentation level, the middle tier, and the database level. In assignments leading up to the capstone project, students are asked to implement technologies apropos to each of these tiers:

- **Presentation layer.** The client's web browser functions as a thin client to create the presentation level. A warm-up exercise asks the student to create a sample e-commerce web site, and then to populate a web form using Java Servlet technology.

- **Middle tier.** Behind the scenes, the middle layer handles low level complex services including security, transaction processing, thread pooling, and resource sharing. Middle tier considerations include the choice of product, for example, J2EE vs. .NET. In our course, students submit a report comparing these technologies, and another report discussing the benefits and pitfalls of open-source software.

- **Data level.** Here, an assignment has the student create and connect to a rudimentary database using Java's `sql` package.

But a comprehensive graduate level client-server course must go beyond simply implementing these three basic tiers. Preparing students for developing real world client-server systems requires that the curriculum address a complex, inter-related, and growing array of theoretical and technical issues. Client-server architecture must be scalable, highly available, maintainable, transactional, and secure, that is, it must be "Enterprise Level". These requirements present the instructor with opportunities to address a number of interesting and subtle issues in software design. Here are some examples of these didactic opportunities:

Efficiency: The designer must insure that as the system grows, the response time remains acceptable. Locality of reference and caching objects that are hard to acquire can improve response time. Enterprise JavaBeans (EJB), known for their transactional capabilities, can also be employed to locally cache non-transactional data (Leff & Rayfield, 2004). This edge-server technique improves performance by moving frequently used information from the data tier closer to the client. Disk affinity means that the more often an instance accesses data, the higher the priority it gains in accessing that data. Performance monitoring may be vital so as to identify problems and repair potential bottlenecks (Andreolini, Colajanni, Lancellotti, & Mazzoni, 2004). A Java Performance Monitoring Toolkit is available to assess middleware performance (Harkema, Quartel, Gijsen, & van der Mei, 2002).

Reliability: It is vital for client-server applications to avoid concurrency problems, yet database systems including `MySQL` do not automatically provide safe concurrency and record locking. The object-oriented paradigm has done much to overcome the inherent challenges of implementing safe concurrency in distributed systems. For example, Stamey and Saunders (2005) provide an elegant prototype for a Java solution to distributed database concurrency implemented using the `AspectJ` extension that adds Aspect Oriented Programming (AOP) functionality to Java. An assignment in this area of client-server computing requires our students to create a threaded server and test concurrent access.

Connectivity: Business-to-business connectivity is also a vital concern. Translation servers can connect to suppliers and interact with their inventory database in order to insure product availability and avoid overstocking the ecommerce site's warehouse. Each supplier requires a specific and different format for sending a request and returns a response formatted its own way. Rather than building a separate XML parser for each supplier, this type of functionality might be implemented using `Xslt` stylesheets. Applying an `Xslt` stylesheet to an XML document generates another XML document with equivalent content but arranged in another form, the form required by the particular manufacturer.

Expandability: As a web-based business grows, it may choose to market new types of merchandise. Ideally, today's data management solution should be fairly easily adaptable to new domains later. This ideal is best achieved by having the middle-tier enforce a clean separation between the front-end client-side architecture and database access at the back-end. The sequence of assignments, from client-side to middle tier to database connectivity, demonstrates this clean separation.

Security: Transactions require secure technology. Encryption protects privacy. Digital signatures can provide nonrepudiation and protect the authenticity, privacy and integrity of e-commerce. An assignment requires students to use Java's `security` package for encryption and digital signatures.

Transactional integrity: Distributed databases are particularly at risk of violating the vital 'ACID' properties (atomicity, consistency, integrity, durability) that are so important in database access. Assuring all of these properties requires precise system design and might include such advanced technologies as write-ahead logging, shadow paging, or two-phase commit algorithms in coordination with the capabilities of the database server. An extension to Java, `Transactional Featherweight Java`, provides transactional capabilities in the form of a "stratified set of rewrite rules parameterized over the meaning of the transactional constructs." (Jagannathan, Vitek, Welc, & Hosking, 2005).

Site promotion: The important business consideration of site promotion can lead to a class discussion of HTML metatags, such as the `KEYWORDS` and `DESCRIPTION` tags.

3. RESULTS

The Nature of the Students' Submissions

Many of our graduate students are professionals. They represent a variety of industries. Therefore, it is not surprising that we received capstone project submissions covering a reasonably wide range of domains in the two capstone experiences. The following executive summaries illustrate a few of the noteworthy projects submitted.

Example 1: "Wonderful Vacations Corporation has been in urgent need of a Timeshare application system to track its sales and generate required legal contract documents. The company uses today an outdated legacy system that has been very difficult to support to due lack of flexibility and functionality. Good Time Share System (GTSS) will develop a solution to attend Wonderful Vacations Corporation today's requirements and will convert existing historical data in to a new database. The project plan satisfies all Wonderful Vacations Corporation requirements. It will be developed using .Net and will use Microsoft SQL Server 2005 database. The plan includes development through completion of a prototype with limited functionality."

Example 2: "This project will address the need of volunteer management software for the Catholic Church of St. Joseph. The software will allow volunteer coordinators to collect, track and administer information to existing, new and potential volunteers, interns, and the agencies that need services. The software will be built using PHP, HTML and SQL, and will utilize an Oracle Database Management system."

Example 3: "Classroom Management of universities is a large, complex, and time-consuming task. Along the time line of each classroom slots must be able to be allocated to instructors and their classes of different departments without violating any predefined rules. The university requires new classroom management database system software to assist users to solve resource-constrained timetabling problem. The university needs a new database to store classroom records and the ability to provide functions to assist a department to acquire timeslots to complete a classroom schedule. This project plan satisfies the requirement with a three-tier application using Java program created within JCreator development environment and a series of Microsoft Access database files. The plan includes development through completion of the first working prototype."

Example 4: This project proposed a client-server system for "Bamazon.com, the world's second largest on-line bookseller". The system utilized the J2EE platform, JSP, Java Servlets, Enterprise Java Bean technology, and MySQL. See Figure 1.
A unique aspect of this project was that it proposed a business model directed solely toward obtaining information for data mining, and then it demonstrated the results of applying the data mining algorithm to simulated data.

The student expressed it thus, "Perhaps the most controversial of our proposals is one that suggests facilitating the customer's acquisition of public domain downloads, even those that may compete with our products. We believe that in addition to cultivating good will, giving potential customers this choice will significantly expand our database. The additional data — otherwise available only from purchasers, a small minority of those who browse the site — will be invaluable in focusing our marketing efforts. Actualizing this recommendation will require major expenditures for a fourth tier of database connectivity and a new stratum of data-mining functionality. In order to support this recommendation, we demonstrate the utility of one data-mining algorithm using our current customer database."

Experience with this Capstone Course

This approach has been used in ten instantiations of the course at 2 universities.

Part of the learning experience was to select appropriate tools and technologies for implementing the final project. The platform chosen by the student was primarily J2EE, due to the Java-centric nature of the overall Masters program. Some who had familiarity with .NET chose to use that platform. A few students based their projects on PHP. The database layer was about evenly split between MySQL, Oracle, MS Access, and MS SQL Server. The project plans were quite ambitious, most estimated by the students as costing over $250,000 for complete implementation.

Student feedback about this course has been quite favorable. Nearly 100% of the students rated the courses as "excellent" or "very useful". One student observed that "The course was a great introduction to software engineering principles." Another student suggested the course "was a comprehensive, well-organized, academically-enriching experience."

The grades on the capstone project were significantly correlated with performance on the more traditional assignments but the correlation was low, representing only 5% of the explained variance. We propose that the traditional text and narrative assignments represent a different skill set from client-server plans and implementation and the incorporation of both activities in our course is beneficial to software skill development.

4. DISCUSSION

Caputo, Kovacs, and Turchek (2006) surveyed information systems professionals in order to learn what didactic areas they perceived as most critical to the workforce. Business information intelligence — "the ability to fuse corporate business goals and objectives with relevant and powerful technological processes" — consistently ranked highest.

Despite the increasing importance of enterprise level systems to business, universities have been slow to fully incorporate client-server computing into their curricula (Seethamraju, 2007). But there are a few noteworthy exceptions. Song, Trajkovskim, and Hong (2004) developed an ambitious model curriculum for a graduate level e-commerce program. Janicki, Fischetti, and Burns (2007) report the successful implementation of a capstone course in which students design systems using advanced technologies such as cascading style sheets and ASP.Net 2.0 to satisfy business requirements of real-world clients. Kovacs and Baugh (2007) report their experience with having upper level students develop three-tiered projects The intent of their project is "not to teach specific skills but to allow the student to merge previously learned skills."

Indeed, several obstacles stand in the way of running an effective client-server computing capstone course. Graduate students enter this course with a variety of backgrounds, some having extensive experience with a given platform, but others having only modest experience with another platform. A single semester course may be adequate time to cover the three basic tiers of a client-server system, but is inadequate time for imparting useful working knowledge about many vital issues such as concurrency, security, expandability and transactional integrity. So, in our approach, we avoid making assumptions about any given student's technological background. We provide a series of exercises designed to move the class toward some common ground, but then we allow each student some flexibility in selecting the domain and applicable technologies for the final project.

Among other insightful observations about educating today's IS students, Saulnier (2007) notes that "The most successful coaching style … has been to ask questions that lead students to formulate their own ideas. Whether in the classroom or in a leadership experience outside of class, this use of inquiry forces students to make the educational experience their own by requiring that they reflect on the challenge at hand and develop a solution of their own."

Our capstone client-server course utilizes small assignments that help the student navigate a number of advanced technologies toward an enterprise-level final project. By the time the students begin designing the final project they have already analyzed and experienced enough technologies that they can effectively select the optimal technologies for their project domain, thus "making the educational experience their own".

We make no claim that that this approach is optimal. But we do believe that it is a reasonable compromise among competing interests, such as: (1) the temptation to try to impart to the student the entire spectrum of skills required for system development versus the limited time actually available for the didactic process; (2) the lure of a project that will afford the student an opportunity to demonstrate some degree of mastery of the design of real world systems versus the comfort and simplicity of more uniform "toy" projects; (3) the extensive, and often specific, background knowledge required for the design of a real world system versus the often large variations in background knowledge possessed by individual students.

Information system development, as taught today, is far beyond what any curriculum would have included just a few years ago. Our approach addresses today's technology and today's needs, and attempts to anticipate some of tomorrow's trends. But, clearly, pedagogic goals must change as technology evolves.

5. REFERENCES

Albanese, M., & Mitchell, S. (1993). Problem-based learning: A review of the literature on its outcomes and implementation issues. *Academic Medicine, 68*(1), 52-81.

Andreolini, M., Colajanni, M., Lancellotti, R., & Mazzoni, F. (2004). Fine grain performance evaluation of e-commerce sites. *SIGMETRICS Perform. Eval. Rev., 32*(3), 14-23.

Barua, A., Konana, P., & Whinstonm, A. B. (2005). An empirical investigation of net-enabled business value *MISQ* March 2005. Retrieved May 26, 2005, from http://cism.mccombs.utexas.edu/works/articles/MISQlast.pdf

Caputo, D. J., Kovacs, P., & Turchek, J. C. (2006). Defining the essential skill and functional areas of study in information technology as measured by a survey of field professionals. *Information Systems Education Journal, 4*(6), 1-8.

Chung, W. S., & McLane, D. (2002). Developing and enhancing a client/server programming for internet applications course. *J. Comput. Small Coll., 18*(2), 79-91.

Foster, I., & Tuecke, S. (2005). Describing the elephant: the different faces of IT as service. *Queue, 3*(6), 26-29.

Gorgone, J. T., Gray, P., Stohr, E. A., Valacich, J. S., & Wigand, R. T. (2006). MSIS 2006: model curriculum and guidelines for graduate degree programs in information systems. *SIGCSE Bull., 38*(2), 121-196.

Harkema, M., Quartel, D., Gijsen, B. M. M., & van der Mei, R. D. (2002). *Performance monitoring of java applications*. Paper presented at the Proceedings of the 3rd international workshop on Software and performance.

Henry, T. R., & LaFrance, J. (2006). Integrating role-play into software engineering courses. *J. Comput. Small Coll., 22*(2), 32-38.

Hutchings, P., & Wutzdorff, A. (1988). Experiential learning across the curriculum: Assumptions and principles. *New Directions for Teaching and Learning (Knowing and Doing: Learning Through Experience), 35*, 5-20.

Jagannathan, S., Vitek, J., Welc, A., & Hosking, A. (2005). A transactional object calculus. *The Science of Computer Programming, 57*(2), 164-186

Janicki, T. N., Fischetti, D., & Burns, A. T. (2007). Incorporating real world projects and emerging technologies into one MIS capstone course. *Information Systems Education Journal, 5*(24), 1-8.

Kovacs, P. J., & Baugh, J. (2007). *Merging object-oriented programming, database design, requirements analysis, and web technologies in an active learning environment*. Paper presented at the The Proceedings of ISECON 2007, v 24 (Pittsburgh): §2722. ISSN: 1542-7382.

Lawler, J. P., Raggad, B. G., & Howell-Barber, H. (2007). Methodology for educating information systems students on the new paradigm of service-oriented architecture (SOA) technology [Electronic Version]. *The Proceedings of ISECON 2007, v 24 (Pittsburgh): §3144*, from http://isedj.org/isecon/2007/3144/index.html

Leff, A., & Rayfield, J. T. (2004). *Alternative edge-server architectures for enterprise JavaBeans applications*. Paper presented at the Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware.

Lefkovitz, D. (2000). The teaching of net-centric computing [Electronic Version]. *In The Proceedings of ISECON 2000, v 17 (Philadelphia): §140*, from http://isedj.org/isecon/2000/140/index.html

Myers, J. P. (2007). A web emphasis in software engineering. *J. Comput. Small Coll., 22*(4), 268-274.

Peslak, A. R. (2006). Web Services: Introduction and Travel Project Tutorials Using Visual Studio and ASP.NET. [Electronic Version]. *The Proceedings of ISECON 2006, v 23 (Dallas): §2334*,

Ramsin, R., & Paige, R. F. (2008). Process-centered review of object oriented software development methodologies. *ACM Comput. Surv., 40*(1), 1-89.

Raoufi, M., Spoa, K., & Wiggins, Z. (2001). Client/server web application development [Electronic Version]. *In The Proceedings of ISECON 2001, v 18 (Cincinnati): §30c*, from http://isedj.org/isecon/2001/30c/index.html

Saulnier, B. M. (2007). "Child is Father to the Man": Social software in the IS 2007 curriculum? . *Information Systems Education Journal 5*(37).

Seethamraju, R. (2007). Enterprise systems (ES) software in business school curriculum - Evaluation of design and delivery [Electronic Version]. *Journal of Information Systems Education*, *18*, 69-84. Retrieved June 18, 2008, from http://findarticles.com/p/articles/mi_qa4041/is_200704/ai_n19432029

Song, Y.-T., Trajkovskim, G., & Hong, S. (2004). Bridging the technological gap between academia and industry:Towards a successful e-commerce graduate program. *Information Systems Education Journal, 2*(9), 1-16.

Stamey, J., & Saunders, B. (2005). Implementing database solutions on the web with aspect-oriented programming. *J. Comput. Small Coll., 20*(4), 249-255.

Stiller, E., & LeBlanc, C. (2006). Teaching software development by example. *J. Comput. Small Coll., 21*(6), 228-237.

Stillman, R. M., & Peslak, A. R. (2007). Teaching software engineering including integration with other disciplines [Electronic Version]. *The Proceedings of ISECON 2007, v 24 (Pittsburgh): §2744*,

Tikekar, R., & Wilson, D. (2001). Implementing an e-commerce curriculum in a CIS program. *J. Comput. Small Coll., 16*(2), 9-20.

U.S. Census Bureau News. (2008). Retrieved June 18, 2008, from http://www.census.gov/mrts/www/data/html/08Q1.html

Yue, K.-B., & Ding, W. (2004). Design and evolution of an undergraduate course on web application development. *SIGCSE Bull., 36*(3), 22-26.

APPENDIX

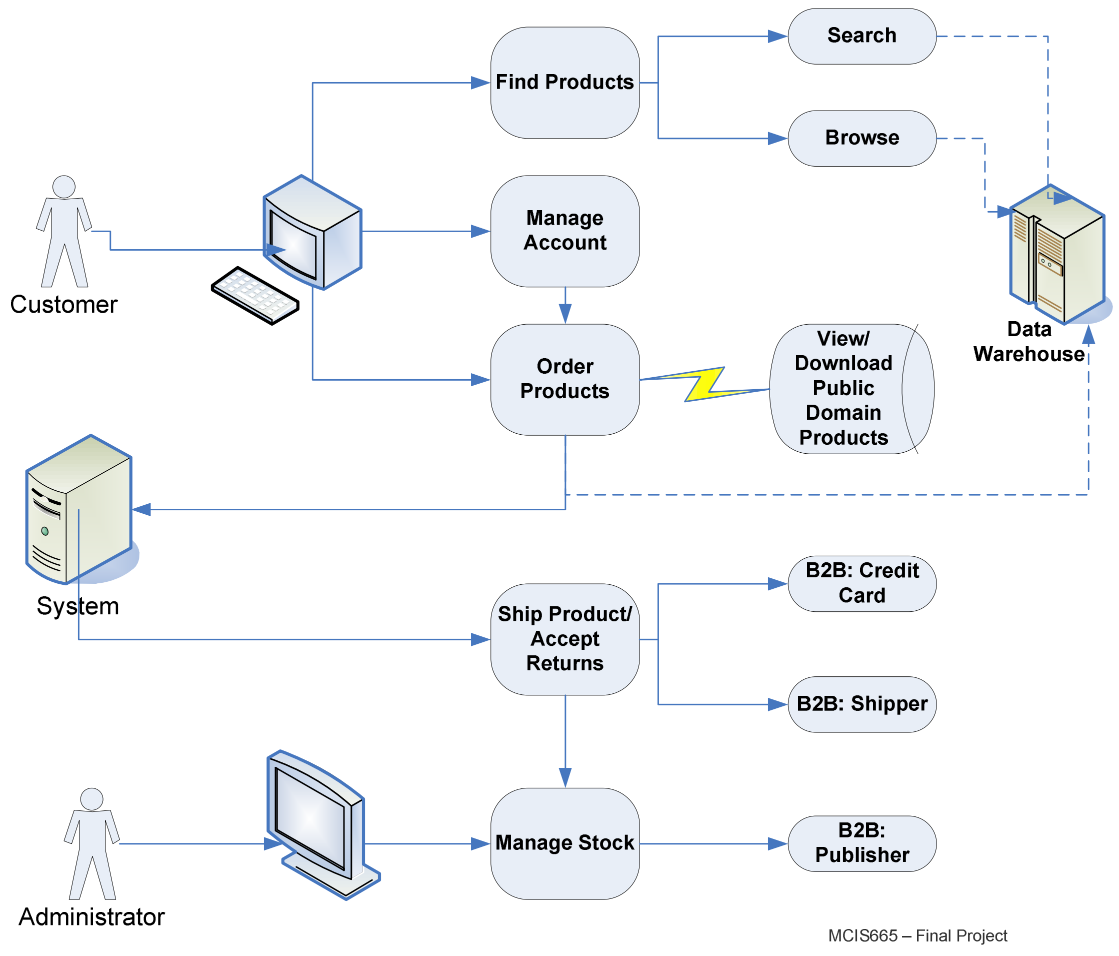

Figure 1. A case diagram of the functionality proposed for the online bookseller of Example 4.