AN INTEGRATED, SYSTEMS ARCHITECTURAL APPROACH TO INFORMATION TECHNOLOGY EDUCATION

Mordechai Gal-Or, Duquesne University, galor@duq.edu
William E. Spangler, Duquesne University, spangler@duq.edu
Trevor H. Jones, Duquesne University, jonest@duq.edu

ABSTRACT

This paper describes a new undergraduate curriculum in Information Technology implemented in the School of Business at Duquesne University. Our approach focuses on the architectural and integration issues related to distributed applications. We argue that the fundamental shift to these types of applications requires that graduates of business IT programs possess the broad set of critical knowledge and meta-skills provided in the new curriculum.

Keywords: curriculum, information, technology, systems, integration

INTRODUCTION

Based on discussions with our industry partners and a thorough review of the academic literature and popular press, we have developed a new curriculum in Information Technology which instills strong technical skills and a detailed knowledge of business systems architectures. While the popular press indicates that companies continue to seek IT employees who exhibit strong 'soft' skills such as inter-personal, teamwork and leadership abilities (1), prior studies of IT employment nevertheless indicate that companies tend to weigh technical skills more heavily (2). Evidence also suggests that companies strongly consider technical knowledge when selecting students for internships – an increasingly popular means of considering students for eventual full-time employment. (3) This is consistent with research showing that companies generally require more technical knowledge in less experienced employees, while emphasizing managerial and interpersonal skills in more experienced employees. (4) Furthermore, a recent study of how IT organizations cope with rapid changes in technology found that the most successful coping strategies included *educating employees on new information technologies* and *retaining employees who are knowledgeable in new technologies*. (5) Thus, we have sought a curriculum that balances a critical understanding of business problems and technology requirements with a strong emphasis and focus on distributed application development through an understanding of current systems architectures.

DESIGNING AN IT CURRICULUM

Curriculum design in the current IT environment requires consideration of two seemingly contradictory issues; i.e., the increasing obsolescence of specific technologies, coupled with the continuing importance of technical skills in new graduates. Westfall suggests that a key to addressing both of these issues is through the teaching of meta-skills. (6) Instead of specific technologies, meta-skills instead focus on the technical abilities necessary to understand and work within a variety of technical environments – abilities such as technology evaluation,

assimilation and debugging. Meta-skills are generic skills that allow students to understand the commonalities inherent across technologies, and therefore adapt to new technologies and environments. The need for IT professionals to avoid obsolescence by engaging in continuous, life-long learning of new skills and technologies has been well-established, and is a principal component of effective professional development programs. (7)

In developing our IT curriculum we essentially have taken a three-pronged approach: 1) determine the *fundamental* (i.e., relatively persistent) transformations in information technology: 2) teach those technologies and concepts in a way that creates and reinforces IT meta-skills: 3) deliver the complete program within a pedagogical environment that stresses the meta skills as a function of the response to business needs, problems and solutions. Our initial exploration of the marketplace, as well as the types of positions and skills of our graduates, indicated that our most successful graduates possessed a strong grounding in business systems – particularly distributed, integrated systems. This initial conclusion was confirmed and fine-tuned through numerous meetings with our industry partners, including companies such as Alcoa, Bayer, Deloitte (now Braxton) Consulting, and FedEx, among others. Our partners helped us to better understand the market forces, the particular trends and pressures these companies face, and the characteristics required of their new hires.

From this analysis, we concluded that the transition toward distributed applications – accessible within and across organizations via various types of systems architectures – is the primary technological transformation affecting the analysis, design and implementation of business systems. It should be pointed out here, that we made a very definitive decision regarding the content and focus of the overall program; while "the network' is a key component of the overall system, it is becoming a separate set of specialized issues and demands a distinct set of skills and knowledge not directly related to our direction toward application development as a response to business problems. Consequently, we have de-emphasized "pure networking" topics, and instead have designed a curriculum to provide students with both a broad and deep understanding of distributed systems applications – particularly those dealing with object-oriented, web-based systems which are quickly becoming the standard paradigm. Toward this end, we sought to encourage students to explore the fundamental constituents of these systems – including object-orientation, internet interface, language protocols, and database connectivity – via diverse implementations that provide students with variations on specific conceptual themes. Over time, students should begin to assimilate and understand those themes. They should begin to develop meta-skills by solving similar problems in different development environments, using different technologies, and they should begin to think fundamentally in terms of object-orientation and inter-related components. (8)

With these objectives in mind, our curriculum takes an iterative, evolutionary approach that builds on prior learning and requires students to apply that learning in new and more complex situations. We begin by exposing students to fundamental business requirements and potential technology solutions. In terms of creating solutions to business requirements, we go on to cover systems architecture, data and process management, and programming issues. After students have completed those courses, they participate in an *Advanced Application Development* course, in which they begin to use their skills to construct a small, multi-tier application – focusing primarily on the application's interface to a database. This course serves to reinforce previous learning, demonstrate the relevance of that learning, present new technologies, and introduce the concept of component integration and communication. Finally, the *Systems*

Integration course incorporates all of these conceptual and technical skills in a capstone experience where students "put it all together" to develop a distributed (web-based) application – using technologies and approaches most appropriate to the problem at hand.

DETAILS OF THE CURRICULUM

Our systems architectural, application-centric approach entails educating students on the fundamental structure and operation of modern business systems. This includes a standard set of foundation IT concepts as well as the systems integration techniques and practices that use and reinforce those concepts. As illustrated in Figure 1, our curriculum is composed of a set of nine required courses, arranged in a specific sequence intended to facilitate the iterative learning of systems knowledge. Although each course plays a particular role in the development of student systems knowledge and contributes specific knowledge to the courses that follow, the integrative portion of the curriculum essentially is implemented in the last two courses (numbered 7 and 8 in Figure 1). As an aside, while the required courses comprise the essential theme of the curriculum, they are supplemented by a set of electives. The electives allow students to follow specific 'tracks' through the program, and to delve more deeply into particular topics.

Notably, we have made a concerted effort to 'sell' the relevancy and importance of each course as students proceed through the curriculum. Each course, starting with the architecture course (3), begins with a description/review of the overall IT program and an indication of where the individual course fits within the curriculum. The instructor covers the specific contribution of the course to the overall objectives of the program, indicates which prior courses are most important to this course – and why – and highlights how the knowledge gained in the course will be relevant to the courses that follow. We use a standard set of PowerPoint slides for this, which provides a consistent message and 'roadmap' for the students to follow.

Foundation Courses

The foundation courses (courses 1-6) are intended to introduce the essential technologies of distributed applications. The contribution of each course is described briefly below.

Information Systems I & II [1]: exposes students to the nature and use of "tool-box" type applications in business (specifically the MS office suite), as well as standard business systems and the major information processing cycles. The course then introduces the major components of ERP systems with the intention of clarifying how technology supports business requirements. In addition to being an introduction to business technology, these courses are also part of the general business core curriculum and are therefore required for all business majors.

Basic Programming Language [2]: provides students with a set of initial programming skills and concepts (using Visual Basic) and introduces students to the technical constituents of application programs. The course serves the important role of introducing students to the process and logic driven requirements of structured programming as well as the development process itself (syntax, debugging, testing, etc.)

Computer Architecture [3]: This class provides a high level, environmental perspective, for the focus of the program and subsequent class content. The class presents the overall technological infrastructure required to support distributed applications in today's business environment. The

course establishes an important theme of the curriculum – that the nature of systems architectures in place today force modularized application development encompassing multiple systems/ application development environments. Toward this end, the course introduces technologies related to various components of layered architectures, such as the ISO model, TCP/IP, HTTP, CGI, and other common protocols. It also ensures that we cover growing and emerging technologies/ standards (e.g. XML) and their relevance to systems and application development.

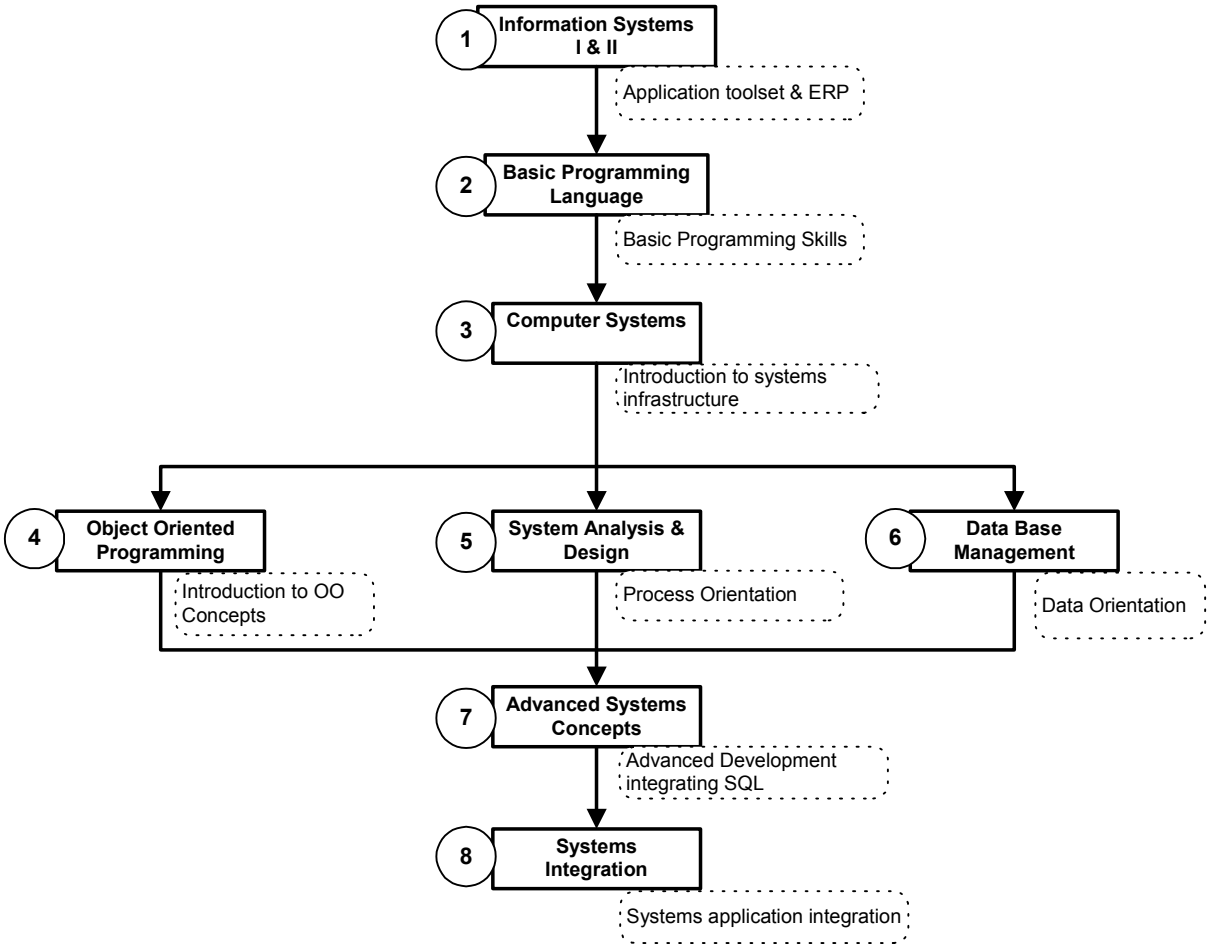

Figure 1: IT Curriculum, showing sequencing of courses and essential contributions

Object-oriented Programming [4]: builds on the introduction to programming, by introducing the principles of object-oriented design and implementation, as well as more complex syntactic structures necessary for more advanced application and solution development. The course emphasizes that the language (syntax) is less important than the inherent logic response to the functionality, in-line with our "meta" approach to education. It currently uses the Java programming language as the development environment.

Systems Analysis and Process Definition [5]: introduces students to a process orientation in systems design, focusing on process modeling tools and techniques, including object-oriented, Unified Modeling Language (UML) constructs. As part of the systems development life cycle,

the course emphasizes the movement from high level (context) models through the decomposition of process definitions to the level of functional program definition. This approach provides a functional, as well as logical, understanding of subsequent application development, which then provides a contextual platform for advanced development classes.

Data Modeling for Database Design [6]: focuses primarily on conceptual business database modeling, leading to logically correct, effective and efficient relational models. Significant emphasis is placed on dealing with complex business data requirements and how the solutions can be modeled and subsequently reflected in the database design. Since the efficiency and effectiveness of functional databases is directly correlated to the structure of the database and not the software platform, physical implementation is *de-emphasized*, in line with our strategy of developing meta-skills and knowledge. Since the de-facto standard for database retrieval is SQL, the course provides an introduction which lends itself to the continuation of SQL development and integration in subsequent courses.

Integrative Courses

Advanced application development [7]: introduces students to the essential practices and tools of distributed application and database development, from an object-oriented perspective. This course focuses on understanding three primary technologies of systems integration: 1) object-oriented development using Java, focusing on data structures, object communication and multi-tier applications, 2) a database management system as a central repository linking distributed modules, and 3) Structured Query Language (SQL) as the primary protocol for communication between applications and databases. While each of these technologies is introduced in the foundation courses, this course extends those technologies in the context of their use in building integrated applications. For example, the Java component explores the use of object-oriented techniques to design and implement distributed applications, primarily through the use of object modules and message passing. Students explore the use of SQL to create, update and query structures in a database, and then learn how to embed SQL within a multi-tier application that will interact with an Oracle server database, using the Open Database Connectivity (ODBC) interface.

The course also addresses the important linkages between the tools and diagramming techniques of object-oriented analysis and design, as specified in UML, and the implementation of UML-based specifications within Java. This approach is mutually reinforcing, in that the UML diagrams help students to better understand the programming code, while the code helps students to better understand the diagrams. It removes much of the abstraction from the diagramming process by adding an element of 'concreteness' to the object-oriented design process. As such, this is a significant aid to students in understanding the precise nature of objects, encapsulation, and message passing – particularly as they are represented in UML representations such as class, use case, and sequence diagrams.

Systems integration [8]: utilizes the concepts and techniques of systems integration as a medium for curriculum integration, with the Internet serving as the context for developing front-to-back integrated applications. The course uses the Internet as an example of a complex, protocol-based distributed environment typically encountered by systems architects, and as such is intended to

provide students with a capstone experience that brings all of the issues of application development together in a project-oriented environment.

The integration course supplements the technical knowledge provided in previous courses with concepts and techniques specific to the task of building internet applications. Thus, students begin by exploring the general nature of web-based applications, using standard 'guestbook' and 'survey' form-based applications as examples. They then begin to examine the basic infrastructure of inter-connected internet applications, including client–server communications between the browser and the web server via the HTTP protocol, as well as the Common Gateway Interface (CGI), which allows application internet programs to communicate with end users through the web server, using the browser as the primary interface. Because the infrastructure is inherently object-oriented, student understanding of message passing and object communication is reinforced through exposure to a conceptually-similar, but physically-dissimilar, development environment.

The integration course also reinforces the concept of application-database connectivity by exploring how CGI programs store and retrieve transactional information in a database – in this case through implementation in a different programming language (i.e., Visual Basic), and using different databases (i.e., Microsoft Access and SQL Server). The issues of application-database connectivity and object-oriented communication then are further generalized through the introduction of additional technologies such as Active Server Pages (ASP) and the ASP object model, as well as the role of Extensible Markup Language (XML) in the object-oriented world of web services.

DISCUSSION AND CONCLUSIONS

Essentially, our curriculum has been structured to discuss and reinforce similar concepts and technologies across multiple courses and contexts. This in turn allows students to understand the commonalities shared by the technologies, and ultimately to acquire valuable IT meta-skills. For example, students generally begin with an introductory course in Visual Basic, proceed on to an introductory course in Java, then develop object-oriented applications using Java, and conclude by developing object-oriented applications using Visual Basic. The use of multiple programming languages to develop distributed applications in multiple contexts is exceptionally valuable. It allows students to generalize from various cases, and by doing so understand the fundamental structure of systems and the tools used to create them. Students begin to understand that Visual Basic and Java have much in common, and with that understanding they begin to develop the meta-skill of adapting to other development tools, many of which exhibit the same fundamental characteristics. Likewise, developing internal applications in Java and Internet-based applications in Visual Basic allows students to recognize the shared characteristics of distributed, object-oriented systems.

The implementation of our new curriculum was greatly facilitated by new resources acquired in part through significant educational grants from the Commonwealth of Pennsylvania. With those grants and considerable internal funding, we were able to construct two new computer labs of 30 to 40 workstations each, as well as a specialized networking lab which allows students to build and experiment with distributed applications. We also have been able to acquire several business applications and other software, including ERP packages from Oracle, SAP and Peoplesoft (through their academic programs). In addition, we have hired new faculty and staff with specialized expertise in systems management, ERP, and database administration,

and currently are training faculty and staff in a variety of technologies related to ERP, database, and modeling tools.

We initiated our new curriculum at the beginning of the academic year, and currently are into the second semester of the implementation. Thus we are still "fine tuning" the content of individual courses and the relationship between courses and their technical prerequisites. While it is too soon at this point to determine the impact of these changes on our graduates, feedback from current students has been useful in helping us to make minor adjustments. We intend to follow up in the near- and medium term with studies of recent graduates and their employers, to determine how the new curriculum has affected opportunities for graduates and their performances within employer organizations. We will be most interested in determining the types of positions our graduates are entering, the companies that are hiring them, starting salaries, expected and potential career paths, and so on. Based on the expectation that rapid changes in technology and business practices will continue, these follow-on studies should serve as input to a continuing process of program definition, course curriculum refinement and improvement.

REFERENCES

1. Dash, J. (2001). Employers say job hunters need soft skills, training. Computerworld, June 4, 2001, 7.
2. Todd, P.A., J.D. McKeen, and R.B. Gallupe (1995). The evolution of IS job skills: A content analysis of IS job advertisements from 1970 to 1990. MIS Quarterly, (17:3), 293-303.
3. Young, D. and S. Lee (1996). The relative importance of technical and interpersonal skills for new information systems personnel. Journal of Computer Information Systems, (36:4), 66-71.
4. Lee, S., D. Yen, and D. Havelka (2001). Evolution of IS professionals' competency: an exploratory study. Journal of Computer Information Systems, (41:4), 21-30.
5. Benamati, J. and A.L. Lederer (2001). Coping with rapid changes in IT. Communications of the ACM, (44:8), 83-88.
6. Westfall, R.D. (2000). Meta-skills in information systems education. Journal of Computer Information Systems, (40:2), 69-74.
7. Schambach, T. and J.E. Blanton (2002). The professional development challenge for IT professionals. Communications of the ACM, (45:4), 83-87.
8. Nelson, H.J., D.J. Armstrong, and M. Ghods (2002). Old dogs and new tricks. Communications of the ACM, (45:10), 132-137.