# Meeting The Challenges Of Today's IT Labor Demands By Building A Better Prepared Student Thru Targeted Curricula Enhancements

Jeffrey L. Brewer, (Email: jbrewer@purdue.edu), Purdue University
Kevin C. Dittman, (Email: dittmank@purdue.edu), Purdue University

## ABSTRACT

*The landscape of the United States information technology (IT) workforce is evolving due to a variety of factors including global outsourcing, commercial-of-the-shelf software (COTS), and the bursting of the dot.com bubble. In addition the world of software development and systems integration continues to evolve into an object-oriented (OO) based environment. The advent of new development platforms such as Microsoft's .Net and Sun's J2EE has accelerated the adoption rate of OO based tools and the object-oriented development paradigm. Tomorrow's workers need to be equipped to describe a world made up of objects – whether the objects are self created or inherited from an existing object framework. In addition to offering OO software development classes, educational institutions must also begin educating systems analysts to create informative models based on OO principles. The Unified Modeling Language (UML) has emerged as the standard for defining modeling principles and techniques. This paper describes how we have tailored our curriculum to meet the demands of our graduates' future employers by offering a four course sequence in systems integration and project management at the undergraduate level. This series of classes that teach students how to elicit requirements and turn them into analysis and design level artifacts based on the UML standard. The classes also teach and use a development methodology which is less methodical and more contingency (agile) based. We also introduce the students to using analysis and design patterns. Course models and examples of learning assessments are included. The paper also maps the learning outcomes from our classes to those purposed in the ACM/SIGITE Information Technology model curriculum.*

## INTRODUCTION

During the past few years, the dynamics of the IT job market have changed tremendously. Even though the overall size of the American IT workforce has grown slightly from 10.3 million workers to 10.5 million workers, the demand for IT workers has actually dropped, from 500,000 jobs filled last year to only 230,000 jobs expected to be filled this year. Of those disciplines affected most by this decline is software programmers. Their numbers represent the largest group in the IT workforce but the demands for programming jobs have declined by almost 30,000 jobs. (ITAA 2004). The reasons for the demand decline are numerous and many are subject to debate, but a few stand on their own merit.

They include:

- The affect of global outsourcing,
- The bursting of the Dot.com bubble,
- The reliance on commercial of the shelf software (COTS).

Studies indicate that since 2000, 100,000 IT jobs have been moved offshore and that between 2003 and 2008, one out of every two new IT jobs will be moved offshore. (ITAA, 2004) Forrester Research of Boston says about 3.5 million U.S. jobs will get shifted to lower-wage, developing countries by 2015. Of those jobs outsourced, the programming sector has been greatly affected (Kranz, 2004). Between March 2001 and November 2003, 313,000 IT jobs were lost because of the demise of the dot.com phenomenon (ITAA, 2005). Again the programming sector was greatly affected. According to Marc Cecere at Forrester Research Inc., the following strategic staff functions should not be outsourced: project management, architecture, planning, vendor management, security policy, business analysis, and project ownership (Melymuka, 2004).

And finally, more and more companies are relying on shrink-wrapped solutions to fill their software and system needs. In some companies the ratio has been as high as 85% purchased software versus 15% custom built to satisfy their IT needs. Thus reducing the demand for software builders and in turn increasing the demand for software integrators. Those individuals who understand how to solve IT problems by integrating purchased software into their existing IT infrastructure often requiring the integration with legacy systems.

Our faculty has chosen to meet the challenge by building curricula that prepares students to be system integrators. We created a curriculum that is very rigorous and robust in systems analysis and design and is focused on the object-oriented approach to system development and the use of the unified modeling language (UML). In addition, the curriculum pays special attention to soft skills; the ability to work in teams, written and oral communication, conduct presentations and meetings, prepare plans, and even manage projects. We feel that with the skills and a solid foundation in systems analysis and design, our graduates are given a competitive edge in the declining job market. Given the trend of reduced IT training budgets (Kranz, 2004) and the notion that companies don't typically outsource all IT functions, for example "companies prize IT pros with the ability to define goals, track their progress, negotiate contracts, and communicate effectively with customers (Krantz, 2004), organizations are looking for prized individuals who have already been educated to be immediately productive and "hit the ground running."

Our curriculum consists of a four course sequence specialization in systems integration and project management. The first course, a requirement for all incoming freshman, begins with a global introduction into the world of a systems analysis and how they use traditional modeling methods (data flow diagrams, entity relationship diagrams, etc.). The second course introduces OO concepts and modeling tools and techniques and the third course focuses on advanced OO topics. The final course introduces the students to the world of project management and the processes and techniques used to manage IT projects today. This paper will describe in detail the topics covered and learning outcomes expected for each of the four courses.

## OOAD DEFINED

As the world of Java and .NET has emerged, experience with the object-oriented paradigm is becoming increasingly important. In a study by the Gartner Group (2003), "Microsoft and Java technologies will command 80 percent or more of new e-business application development initiatives through 2008." Doug Falk, CIO of National Student Clearinghouse, pays a 10 percent premium on hires with multi-tiered object based architecture experience and web services experience (Johnson, 2003). Understanding objects and object based tools has been proven essential for IT workers to continue to be successful while these environments become the current and future of Information Technology (IT). Downes (2003) selected object-oriented design tools as one of the seven pillars defining the next generation of architectures.

Object-oriented analysis and design (OOAD) is based upon the idea of objects and object modeling. Ambler (2001) defines an object as a person, place, thing, concept, event, screen, or report. Objects contain both data and functionality. Object modeling is a technique for identifying objects within the systems' environment and the relationship between those objects. Object modeling allows for an enhanced view of how objects interrelate inside the system as well as a simplified development process.

To further understand OOAD, one must have a better understanding of the process used to develop information systems. A methodology formally defines the process that you use to gather requirements, analyze them, and design an application that meets them in every way (Object Management Group, 2003).

A methodology can also be defined as the physical implementation of the logical life cycle that incorporates the following characteristics:

- Step by step activities for each phase
- Individual and group roles to be played in each activity
- Deliverables and quality standards for each activity
- Tools and techniques to be used for each activity

The methodology that is becoming most commonly used with object-oriented development projects and one that we teach to our students is an agile iterative/incremental approach (Ambler 2001). An iterative/incremental approach will include multiple development products built and released to the users in small releases, a portion at a time, until ultimately meeting the complete project requirements. The goal of an iterative/incremental approach is to allow for mid project changes, to show signs of progress earlier, and deliver functionality to the user and receive feedback sooner. This new method attempts a useful compromise between no process and too much process, providing just enough process to gain a reasonable payoff (Fowler, 2003). The best known and some of the earliest work on this methodology came from Boehm's (1988) spiral approach.

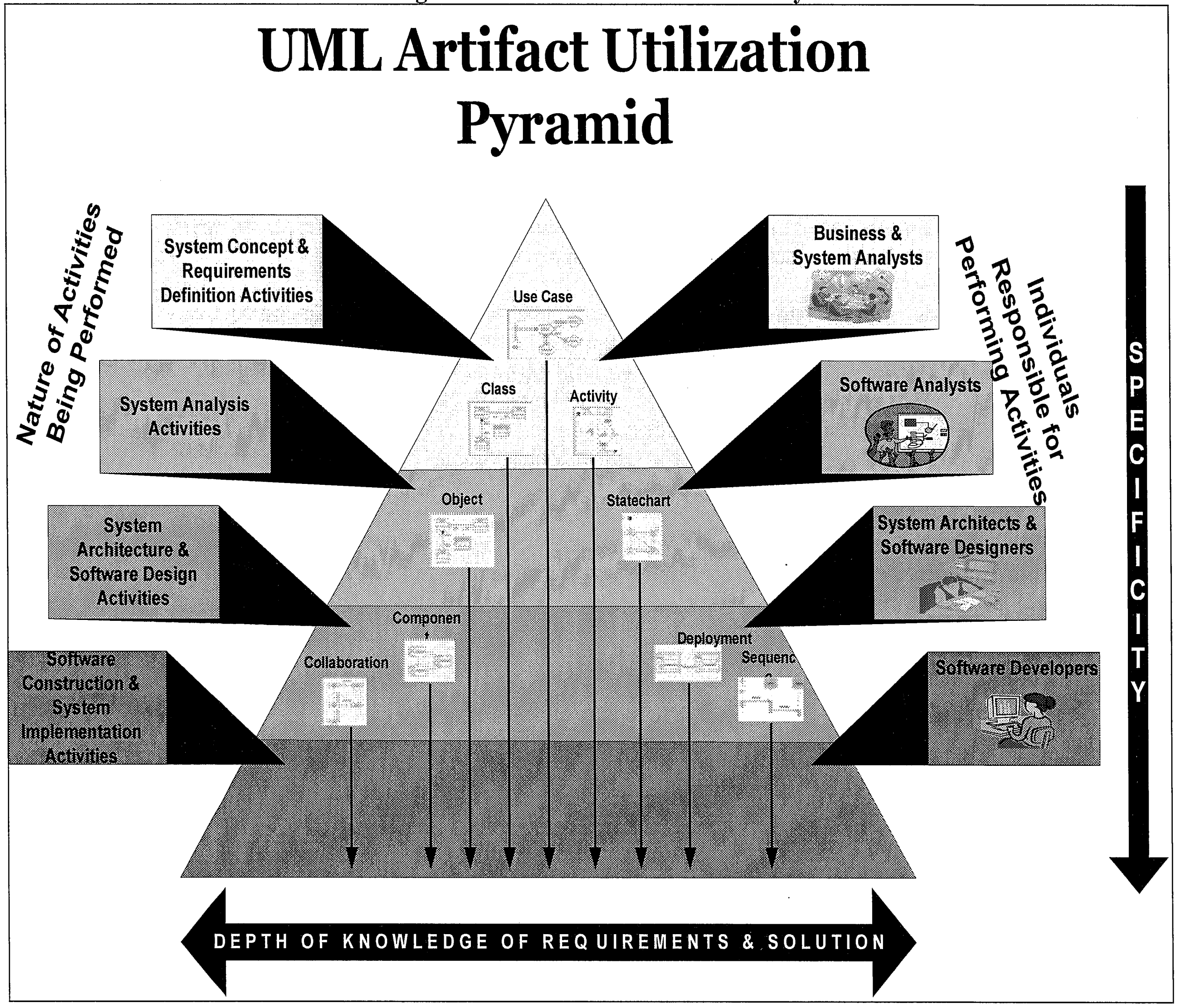
The Unified Modeling Language (UML) helps you specify, visualize, and document models of software systems, including their structure and design, in a way that meets all of the requirements of the system. Using the UML and its tools as a guideline, future application's requirements can be analyzed and a solution can be designed that meets them (OMG, 2003). The UML is a set of modeling tools and techniques that was founded by three pioneers of the object-oriented paradigm. Grady Booch, Ivar Jacobson, and Jim Rumbaugh incorporated their ideas and strengths to develop the UML and submit it to the Object Management Group for standardization in 1997. UML version 2 was just released and adopted in early 2005.

The UML v2 consists of thirteen types of diagrams. "A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and paths (relationships)" Booch, Jacobson, Rumbaugh (2005). The diagrams are as follows:

- Class diagram
- Object diagram
- Composite Structure diagram
- Use Case diagram
- Communication diagram
- State diagram
- Activity diagram
- Sequence diagram
- Component diagram
- Deployment diagram
- Package diagram
- Timing diagram
- Interaction overview diagram

Figure 1 is a model of the key UML diagrams we emphasize in our curriculum and how they are utilized during the execution of a system development project. At the top of the pyramid we begin the project and over time move toward the bottom of the pyramid. At each stage signified by the sections of the pyramid we learn more about the requirements of the project and add more detail to each of the models. As we move through the life cycle, different models take on more significance. For example, at the beginning of the project we employ use cases to capture customer requirements in a style that is easy to write and easy to be understood by both technical and non-technical individuals. You will notice in Figure 1 that under use case that a line extends to the bottom of the pyramid signifying that they are used throughout the project. The larger downward facing arrow labeled "Specificity" on the right hand side of the model tells us that each of the models take on more and more detail as we work through the project. The use case we started with takes on more and more detail as we discover more about the user's requirements, the architecture of the deployed system, and the other computer systems that it will need to interact with. Also on the UML Artifact Utilization Pyramid on the right had side are the principle individuals who are involved in leading the effort to perfect the UML models at each stage of the process and on the left hand side are the labels for each of the major phases.

**Figure 1: UML Artifact Utilization Pyramid**


UML Artifact Utilization Pyramid
Nature of Activities Being Performed
System Concept & Requirements Definition Activities
System Analysis Activities
System Architecture & Software Design Activities
Software Construction & System Implementation Activities
Use Case
Class
Activity
Object
Statechart
Componen
Collaboration
Deployment
Sequenc
Business & System Analysts
Software Analysts
System Architects & Software Designers
Software Developers
Individuals Responsible for Performing Activities
SPECIFICITY
DEPTH OF KNOWLEDGE OF REQUIREMENTS & SOLUTION

In our series of object-oriented classes, each type of UML diagram is touched upon, however, as mentioned earlier there are several that are emphasized, these being the Use Case diagram and corresponding narrative, Sequence diagram, Class diagram, Component diagram, and Deployment diagrams. These models were chosen based on the authors combined 45 years of industry experience in the IT industry and feedback received from our industrial advisory board. The Use Case diagram provides a basis to help identify objects and their high-level relationships and responsibilities within the system. The Class diagram takes the objects found in the use case diagram and depicts the system's object structure. The Component and Deployment diagrams involve the implementation of a system. The Component diagram shows the software components that make up a larger piece of software, their interfaces, and their interrelationships (Ambler, 2001). The Deployment diagram depicts the hardware, software and middleware configuration of a system. Sequence diagram creation is also studied in the course. These diagrams are important because they graphically represent how objects interact with each other via messages in the execution of a use case or operation. We spend a great deal of time in class focusing on these diagrams to understand the fundamentals of the UML and how they (the diagrams) all work together to break down the complexity of systems and make them more understandable for users, system developers, and analysts.

## REVIEW OF CURRENT CURRICULUM

Our Information Technology curriculum is divided into four separate emphasis areas, software development, systems integration, data management, and network engineering technology. The students are required to declare an emphasis area by the end of their 4th semester. With the exception of one class in our networking area, all students take the same core set of classes during their first four semesters. The students are allowed to move from one area to another but are required to take all required courses in the chosen emphasis area. In the following paragraphs are abbreviated descriptions of the different areas taken from our department website. Team based instruction techniques are used in each to enhance the student learning experience. The students are introduced to system development lifecycle methodologies and object-oriented modeling concepts using the UML during their very first semester. A thorough understanding of OOAD tools and techniques is done in two subsequent courses discussed later in this paper.

In the Software Development area, students learn how to apply contemporary technologies and techniques to develop and maintain all levels of application software. The computing courses focus on using programming languages to construct these software applications for a variety of hardware and software platforms and networks. To complement the computing courses, courses in interpersonal communications, business, and liberal arts are also required.

In the Network Engineering Technology area, students learn how to design, construct, troubleshoot, and manage sophisticated voice, video, and data networks. This unique curriculum emphasizes data, image, and voice communications using telecommunications technology. The networking courses focus on subjects such as digital communications, local and wide area network design, wireless networks, system administration, network security, digital forensics, and network planning and management. The ability of graduates to communicate with application and database professionals is enhanced with courses in application development, database design and administration, and system analysis. To complement the computing and telecommunications courses, courses in interpersonal communications, business, economics, liberal arts, and electrical engineering technology are required.

In the Data Management area, students learn how to analyze, design, construct, and implement sophisticated database and data warehousing systems for business transaction processing and operations, information management, and decision support. The technology courses focus on subjects such as database analysis and design, data and database management, database programming, and database administration. To complement the technology courses, courses in interpersonal communications, business and liberal arts are required.

In the Systems Integration (SI) area, students learn how to develop and integrate unique information systems solutions such as e-business, enterprise applications, and all types of information systems into a business. Emphasis is placed on systems thinking and problem solving. Technical courses emphasize systems analysis, systems design, IT hardware and software procurement, outsourcing, prototyping, and application development, systems integration, systems implementation and project management. To complement the computing courses, courses in interpersonal communications, liberal arts, and either business or manufacturing are required.

## SYSTEMS INTEGRATION COURSE DESCRIPTIONS

In this section of the paper, we describe the tools and techniques used to obtain the learning outcomes desired from students after they complete each of the four systems integration specialization courses. The learning outcomes in this paper are linked with outcomes documented from the Association of Computing Machinery Special Interest Group on Information Technology Education (ACM/SIGITE) Information Technology curricula (SIGITE, 2005). We have not attempted to list every outcome but a select list of important concepts. Some of the key areas addressed are systems thinking and analytical thinking, computer application systems, object-oriented analysis and design, computer network architectural overview, communication skills, problem solving, systems development methodologies and interpersonal skills, and managing IT projects.

The learning objectives of the four courses discussed in this paper, 180 - Introduction to Systems Design, 280 - Systems Analysis and Design Methods, 380 - Requirements Discovery and Modeling, and 480 - Managing IT projects, are aligned with the following SIGITE 2005 body of knowledge core topics:

- IM – Information management {IM3 Data Organization Architecture and IM4 Data Modeling}
- SIA – System Integration and Architecture {all sub categories}
- SP – Social and Professional Issues {SP1 Professional Communications, SP4 Teamwork Concepts and Issues, and SP7 Organizational Context}

The study of different computer application systems is done throughout the four courses. In the first course, 180, students are given an introduction to the following topics:

- types of information technology systems,
- systems development methodologies and life cycles,
- database management systems and logical data modeling,
- UML and object-oriented systems,
- Process modeling using data flow diagrams (DFD),
- Requirements discovery

This course surveys the entire systems development life cycle. Students learn terminology and begin to build simple models (UML and DFDs) based on requirements from small case studies. The students are also required to build a very simple working software prototype which satifies a list of requirements. Labs emphasize data and process modeling and data querying using SQL to prepare students for later systems, programming, and data management classes.

In the second course, 280, all of the topics from the first course are covered once again in more depth. In the first course students create very simple UML models but mostly concentrate on learning the terms and concepts and how to read UML models. The second course students are given several opportunities to build UML models based on larger more involved case studies. The concepts of newer agile systems development processes are discussed. Whitten (2004) defines a systems development process as "a set of activities, methods, best practices, deliverables, and automated tools that stakeholders use to develop and maintain information systems and software." The students learn to apply several different tools and techniques in the development of a complete life-cycle methodology. The students learn more about different types of application systems: transaction processing, management information, decision support, expert systems, office automation and work group computing. An introduction to user interface design concerns is covered. The students also begin to learn about multi-tiered architecture environments, where they apply and how to model them. The second course also contains a team based project. In the next course, 380, students work on building analysis and design level documents using an iterative incremental systems development methodology. During 380 the students are building elaborate UML models and other documentation which takes into consideration enterprise level application, data, and process issues. Additional topics include analysis and design patterns, middleware, application and data partitioning, and legacy system integration techniques. Labs for both 280 and 380 focus on UML model creation using IBM's rational rose and requirements management using IBM's Requisitepro.

The students are provided case studies based on real-life projects worked on by current faculty. The third course, 380, uses these cases studies to create detailed artifacts concentrating on the early phases of a systems development project – namely requirements discovery, analysis, and design. Each case study emphasizes the importance of understanding how the new proposed systems development effort – either build from scratch or an off the shelf solution – must interact with the current legacy environment. We are graduating systems integrators with integration skills along with traditional systems analysis skills. Integrators must understand how each part of the current system will be affected by something new. They must use a "systems approach", previously defined in this section, to understand the impact on the company as a whole.

Student communication skills are enhanced by requiring them to work in teams to complete assignments and to generate detailed technical documentation. The students work in teams of three to five members. The students must collaborate in order to complete the assignments. Each student is required to fill out a confidential team peer evaluation at the end of each milestone to explain and rank each member's contribution to the group project. The students are also required to present their work to the class. Each member of the team is required to speak. The audience is made up of other class members, graduate teaching assistants and the professor. The audience is required to fill out a presentation review form to explain how the team presenting might improve their presentation skills and what issues they had with the business solution presented. The students are also assessed on their ability to explain in writing complicated technical information suitable for end-users and management to understand. The papers are graded based on writing ability as well as on content.

To prepare students for a world that has legacy systems that must be communicated with, large full scale ERP implementations, and mixed new and old architectures, the students must be presented with a project which forces them to think about the entire enterprise. The students first must understand the existing legacy systems, what information is required and how to interact with them via middleware tools. The internet has also added a layer of complexity to the systems integrator's job. They must understand how and when to expose their applications to the World Wide Web.

All three classes discussed in this paper require projects be delivered by students working in teams to help them develop their interpersonal relationship skills. The students must communicate and coordinate with the group to accomplish the goals of the homework assignments. The curriculum requires all students to take classes in business writing, technical writing and communications. In our analysis and design classes we give them the opportunity to use these skills. The students must use their business and technical writing skills throughout all of their course work. To help accomplish this we vary the intended audience of the assignments, some directed toward business users and some directed toward other technical professionals.

In the "real-world", systems integrators are required to obtain user requirements in a variety of ways. Because of this, we have tried to force students to learn to break down the large problem presented in the case study into manageable pieces to understand the problem completely and be able to offer a viable solution. The case studies present information in several different formats: user dialogue, background narrative, list of issues, and others to teach students how to do requirements discovery. The students are forced to interrogate the entire case study for answers, they must ask their team mates questions and then must also approach the Professor asking questions to help fully understand the entire business problem.

Students need to have a complete tool box of techniques in order to build today's complicated systems. It is important for them to understand, that in many of the newer methodologies, most artifacts are optional, except maybe the actual software code. Larman (2002) uses the analogy of a set of medicines in a pharmacy. People shouldn't randomly take medicine but should match the medicines they take to the ailments they have.

Students and practitioners must learn to pick the right tools to match the situation:

- A "heavy" methodology versus a "light" methodology,
- Correct timebox time line,
- Traditional modeling tools versus object-oriented modeling tools,
- Define which artifacts should be persistent and maintained.

A key ingredient in developing students into systems analysts and systems integrators is the understanding of systems thinking -- the ability to see how the parts fit together to make the whole organization function. Students must understand organizational structures and information technology's role in the enterprise. They must understand that each project affects more than just the immediate connected applications. We address this with our case studies and in class examples. The students must balance their project objectives with those of the organization and its stakeholders. Their solution must interface with outside entities as well as in-house legacy systems.

Finally, the fourth course in the sequence is managing IT projects. The material for this course is based on the project management body of knowledge defined by the Project Management Institute (PMI). PMI is the largest organization world wide in this space with 200,000 plus members. The students learn the difference between the project management lifecycle and a product lifecycle. This course introduces the application of knowledge, skills, tools, and techniques that project managers use to plan, staff, estimate, and manage information technology projects. Special emphasis is placed on learning and applying the concepts of managing scope, risk, budget, time, expectations, quality, people, communications, procurement, and externally provided services. Students will apply project management technology and techniques to real business problems.

## FUTURE COURSE DIRECTIONS

We are moving more and more OO concepts into the freshman course. From experience of teaching these courses for several years, we have found that the students need repeated exposure to many of these concepts before mastery is obtained. Students in our curriculum begin from day one learning an OO based software language. We feel it is extremely important that they also start learning to create and interpret OO based models. Another reason to move OO earlier in the curriculum is to support our upper level software development courses. Teachers in these courses are beginning to write homework problem statements, exam questions, etc., using UML notation. We are also responding to our industrial advisory board which has specifically asked us to include more OO based learning into our curriculum to better prepare students.

We are also in the process of upgrading all of our labs with state of the art case tools. Case tool technology has improved significantly in the last several years in the support of UML model creation and the forward/reverse engineering of OO programming languages. We use lab sessions to teach the students how to use these case tools and require that all homework submissions be generated from these tools. This becomes important so that our students not only understand OO concepts, but can also apply them with the case tool technology.

## CONCLUSION

The world of IT is a constantly changing place. Students must learn to use the tools of today and be prepared to learn the tools of tomorrow. Our curriculum and more specifically the four courses mentioned in this paper give students a thorough understanding of today's business environments and tools (UML, iterative/incremental development, OOAD, and project management). They also build in them a foundation to explore and obtain new knowledge.

**REFERENCES**

1. Ambler, Scott. (2001). *The object primer: The application developer's guide to object orientation and the UML*. Cambridge: Cambridge University Press.
2. SIGITE (2005). Computing Curricula Information Technology Volume. Retrieved October 14, 2005 from http://sigite.acm.org/activities/curriculum
3. Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, pp. 61-72, 21(5).
4. Booch, G., Rumbaugh, J., and Jacobson, I. (2005). *The unified modeling language user guide*. UpperSaddle River, NJ: Addison-Wesley.
5. Downes, Larry (2003). Unleashing killer architecture: the shape of things to come. Retrieved June 15, 2003 from http://www.cio.com/archive/061503/architecture.html
6. Driver, Mark (2003). J2EE and .NET for e-business development efforts. Gartner Group. Retrieved June 17, 2003 from http://itap.purdue.edu/itresources/gartner/research/114600/114609/114609.html
7. Fowler, Martin (2003). The New Methodology. Retrieved June 30, 2003 from http://www.martinfowler.com/articles/newmethodology.html
8. Information Technology Association of America. (2005). The Comprehensive Impact of Offshore Software and IT Services Outsourcing on the U.S. Economy and the IT Industry. Retrieved November15, 2005 from http://www.itaa.org/.
9. Information Technology Association of America. (2004). Adding Value...Growing Careers. The Employment Outlook in Today's Increasingly Competitive IT Job Market.
10. Retrieved November15, 2005 from http://www.itaa.org/.
11. Johnson, Amy H. (2003). Preparing for a Career in Web Services. Retrieved June 17, 2003 from http://www.computerworld.com.
12. Krantz, Garry (2004). Outsourcing Trend Touches Independents Too. Retrieved November 18, 2005 from http://searchcio.techtarget.com/originalContent/0,289142,sid19_gci1006375,00.html.
13. Larman, Craig (2002). *Applying UML and patterns: an introduction to object-oriented analysis and design and the unified process*. Upper Saddle River, NJ: Prentice Hall, Inc.
14. Melymuka, K. (2004). Sidebar: The Keepers. Retrieved September 19, 2005 from http://www.computerworld.com/careers/story/0,10801,95945,00.html.
15. Object Management Group. (2003). Introduction to OMG's Unified Modeling Language. Retrieved June 17, 2003 from http://www.omg.org.
16. Satzinger, John W., and Orvik, Tore U. (2001). The Object-Oriented approach; concepts, system development, and modeling with UML. Boston: Course Technology.
17. Whitten, Jeffrey L., Bentley, Lonnie D., and Dittman, Kevin C. 2004. *Systems analysis and design methods*. New York: McGraw-Hill.

**NOTES**