Journal of Information Systems Education, Vol. 17(3)

Contemporary Approaches and Techniques for the Systems Analyst

Dinesh Batra
Decision Sciences and Information Systems
College of Business Administration
Florida International University
University Park, Miami, FL 33199
batra@fiu.edu

John W. Satzinger
Computer Information Systems Department
Missouri State University
901 S. National Avenue
Springfield, MO 65804
JohnSatzinger@MissouriState.edu

ABSTRACT

A recent survey of methodologies and techniques currently used in organizations for developing information systems indicates significant trends that call for a revision of the Information Systems (IS) Systems Analysis and Design (SA&D) course to define what methodologies, techniques, models, and tools need to be taught. As authors of analysis and design textbooks, we are particularly concerned about these trends, as are all who are involved in information systems educational programs. Each program needs to consider how to incorporate three fundamental changes on the SA&D curriculum – the growing popularity of object-oriented techniques, the emergence of the iterative approach, and the increasing adoption of the agile approach. This article discusses these three fundamental changes and references research describing the recent trends. Based on this research and on our experience teaching and writing about analysis and design, we make some recommendations. Given the vast number of topics in analysis and design, it is time to seriously consider including two courses in the IS curriculum that can deal with the breadth of the system related topics in the contemporary environment. In terms of functional requirements and analysis issues, we argue for employing a use case driven approach. We recommend that the SA&D courses use Unified Modeling Language (UML) whenever possible for modeling; however, we note some of the usability problems of UML. We suggest that the time has come to drop the data flow diagram (DFD). We also consider the impacts of the outsourcing trend on the course coverage.

Keywords: Trends in Analysis and Design, Analysis and Design Techniques, Teaching Analysis and Design

1. INTRODUCTION

Recent and substantial developments in systems analysis and design methodologies and techniques have probably affected the need for a significant revision of the IS Systems Analysis and Design (SA&D) course. After many relatively stable years emphasizing a waterfall system development life cycle (SDLC) using structured analysis and design modeling techniques, the analysis and design course gradually evolved in the 1990's to add more emphasis on data concepts, interactive interfaces, prototyping, client-server systems, enterprise systems, and more recently Web-based technologies. Additionally, object-oriented technologies and techniques as well as agile development methodologies have emerged to address such trends. Thus, it is time to re-examine the systems analysis and design course in the IS curriculum and to define what methodologies, techniques, models, and tools need to be taught in the IS Systems Analysis and Design course.

Most MIS degree programs have just one SA&D course (Gorgone et al., 2002; Gorgone et al., 2006). The typical SA&D instructor faces a number of difficult questions when trying to fit the much larger range of topics into a single course. How does one fit the structured, the iterative, and the agile approaches in one course? Should the life cycle notion be taught as iteration to reap the advantages of the structured and iterative approaches? Can agile principles be included in the same course even though there are some fundamental differences with the structured and the iterative? Is it time to

drop the data flow diagrams (DFD's)? Should the use case be the basic unit of requirements documentation? Can use cases be employed in a structured approach? Can data modeling be used along with object-oriented techniques? How many UML diagrams need to be covered? How do we reconcile object-oriented development with relational systems? Should the focus of the course be toward web-enabled systems? Given the outsourcing environment, how much design and implementation should be covered as compared to requirements gathering, analysis, and project management? Should the MIS program consider a second course to fit the topics? Or, should the SA&D instructor work with the instructor who teaches the project management (PM) course, which can incorporate the management principles of the three systems development approaches?

In this paper, we attempt the arduous task of finding reasonably satisfactory answers to such questions faced by the SA&D instructor. In our opinion, there is no optimum solution; we provide guidelines and recommendations that can be adapted based on specific privileges or constraints of the MIS program.

We employ the terms approach, methodology, technique, model, and tool, terms which are sometimes used interchangeably in the literature. So it is appropriate to provide a working definition of each of these terms. An *approach* refers to an essential systems analysis and design philosophy. Currently, there are three popular approaches – structured, iterative, and agile. A *methodology* is an instantiation of an approach, e.g., Rational Unified Process (RUP). Sometimes a methodology is called a method, although we try to avoid the use of the latter term. A *model* is a formalism and a way of representing important constructs. A *technique* denotes how the model is used. An ER model (Chen, 1976) can be described as a model or a technique depending on the context. As a model, ER provides the constructs (entity, relationship, attribute, generalization). As a technique, ER provides a procedure for data modeling (e.g., see Teorey, Yang, and Fry, 1986). A tool is software support for a technique or a methodology. We admit that there is overlap among these terms, and that the usage sometimes is ambiguous.

2. WHAT IS THE FUNDAMENTAL CHANGE?

A recent survey of methodologies (Lang, 2006) currently used in organizations for developing web-based systems shows that the notion of methodology in the traditional systematic sense seems to have been largely displaced by hybrid aggregations of techniques and other method fragments. These aggregations and fragments are selected on the basis of usefulness and purposefully blended within the overarching framework of an in-house development process. The survey reported that the use of methods is 23% for hybrid, 22% for structured, 15% for agile, 14% around tools, 13% for iterative/incremental, and 8% for object-oriented, among others (see Table 1). Note that the adoption of object-oriented development is low because models and techniques involving UML are not standalone methodologies. It might be better to view object-orientation in the realm of techniques.

The use of models/techniques is 95% for flowcharts, 74% for entity relationships models, 72% for use case diagrams, 62% for class diagrams, and 50% for state machine diagrams, among others (see Table 2). It can be deduced that the use

Approach	Adoption
Hybrid, customised, or proprietary in-house method or approach	23%
Traditional "legacy" software development methods and approaches, or variants (*e.g. SSADM, Yourdon, JSP, SDLC / Waterfall*)	22%
Rapid or agile development methods and approaches (*e.g. RAD, Extreme Programming*)	15%
Approaches that are focused around the use of tools and development environments (*e.g. PHP, Java, Flash, ASP, J2EE, InterDev*)	14%
Incremental or evolutionary methods and approaches (*e.g. Spiral Model, RUP, Staged Delivery, Iterative Design*)	13%
Object-oriented development methods and approaches (*e.g. OOAD, UML, J2EE*)	8%
No method used / development is "ad hoc"	8%
HCI / Human Factors Engineering methods (*e.g. User Centred Design, Goal-based Requirements*)	5%
Technique-driven development (*e.g. Storyboarding, Flowcharts, UML, Prototyping*)	6%
Specialised non-proprietary methods for Web/hypermedia systems design (*e.g. Fusebox, WSDM, HDM*)	5%

Table 1: Use of Methodologies in Web Enabled Systems (adapted from Lang, 2006)

Technique	Adoption
Screen prototypes / Mockups	97%
Flowcharts	95%
2-D site mapping techniques	91%
Storyboards	85%
Entity-Relationship Diagrams	74%
Use Case Diagrams / Scenarios	72%
Object-Oriented Class Diagrams	62%
3-D site mapping techniques	52%
Statecharts / State Diagrams	50%

Table 2: Use of Techniques for Modeling in Web-Enabled Systems (adapted from Lang, 2006)

case and class diagrams are employed beyond the iterative and object-oriented approaches, and the entity relationship model is used beyond the structured approach. The overall trend is toward employing techniques normally associated with object-oriented development (e.g., use case, class, and state machine diagrams) while retaining certain key techniques from legacy (e.g., entity relationship and flowcharts).

In revising the analysis and design course, the first question to address is how to refer to the fundamental changes affecting the course. Although Lang (2006) does not explicitly compare methods over a time period, the results indicate that there are changes along two dimensions – the techniques, and the methodologies. In terms of techniques, the change is obvious – the predominant increase is in terms of what are normally associated with object-oriented development. Many of the techniques did not exist a decade ago. This is the first fundamental change. It seems that the developers associate "object-oriented" with techniques, but not with a methodology. This is not surprising; for example, RUP is presented as an iterative methodology that employs object-oriented techniques (Kroll and Kruchten, 2003). Some changes are subtle. For example, the use case diagram is widely employed suggesting it not the hegemony of object-oriented development. Overall, the trend is, indeed, toward increasing use of object-oriented (OO) techniques.

There are many sound reasons for defining the shift based on OO techniques. First, the emphasis on OO programming in introductory programming courses makes the use of OO analysis and design techniques the logical direction. Additionally, the most publicized of the UML diagrams refer explicitly to OO constructs (Booch, Rumbaugh, and Jacobson, 1999), so teaching UML class diagrams and sequence diagrams in the analysis and design course is a reasonable direction, although many UML diagrams are not strictly OO (e.g., activity diagram, use case diagram, package diagram). Finally, OO has been the buzzword for newer, better, faster, and smarter, even though the impact is more complicated.

In terms of methodologies, there are varied approaches with none clearly dominating over others (see Table 1), a clear shift from the singular domination of the structured approach until about a decade ago. Many view a second fundamental change as the shift from a waterfall life cycle to an iterative life cycle. A major adjustment is required when describing iterative project planning and project management. Iterative life cycles began with early prototyping methodologies. The spiral model (Boehm, 1988) was the first to formalize iterative development by breaking away from the waterfall life cycle model and focusing on iteration planning based on risk. The Unified Process (UP) life cycle provides a mature iterative approach for project planning and project management (Jacobson, 2000).

Superimposing the first change (techniques) and the second change (methodologies) leads to some interesting combinations. Thus, it is possible to teach SA&D in a sequential, waterfall framework including all of the traditional analysis and design activities and techniques but substituting OO analysis and design models. At the same time, it is possible to teach analysis and design for an iterative, adaptive environment while still using structured analysis and design models.

The third fundamental change is the move from a more formal (disciplined) approach toward a more agile system development. The frustration with the bureaucracy of the disciplined approaches has led to the proposal for agile development (Boehm and Turner, 2004). The new approach is defined by the Agile Manifesto (http://agilemanifesto.org/), which states that developers should value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan (Larman, 2003). Examples of agile development include methodologies such as XP (Beck, 1999; Auer and Miller, 2002), Adaptive Software Development (Highsmith, 2000), Crystal (Cockburn, 2004), and Scrum (Schwaber and Beedle, 2002). Ambler and Jeffries (2002) argue that the Unified Process can be used in a more agile manner with some modifications. Agile methodologies have emerged in response to a dynamic environment, where requirements changes need to be accepted and incorporated. The agile approach focuses on developing working code built through short iterations and

relies on feedback rather than planning as a guiding mechanism. Agile methods require a high level of interaction, collaboration, and face-to-face communication among users and developers. The methods emphasize simplicity, prescribe less documentation, and rely mostly on the tacit knowledge developed through collaboration.

In summary, probably the most fundament question to answer when updating the SA&D course is choosing the most important emphasis for your local needs. It is important to remember that each of these three fundamental changes can be combined together in varying degrees, and there is a need for research toward a framework that more formally defines these changes. The scope of this paper, however, is restricted to pedagogical issues.

3. IS IT REASONABLE TO HAVE TWO ANALYSIS AND DESIGN COURSES?

It is evident that given the magnitude, the three fundamental changes discussed in the previous section are difficult to address in one cohesive analysis and design course. It may be time to argue for a second analysis and design course in the IS curriculum or to incorporate some of the additional material in an existing course. In fact, there are many more issues to address related to requirements determination, analysis, and design (discussed later in this paper). Further, another trend in today's business environment is distributed software development (Coar, 2003; Brown and Wilson, 2005). This trend is more pervasive as a result of globalization, outsourcing, mergers and acquisitions, and the support of advanced communication technologies (Sahay, 2003). In distributed projects, team members are usually located in different places, which may belong to different countries as in the case of off shoring. Distributed environment increases complexity of the software development process due to differences in time zones, geographies, language, and culture (Herbzleb and Mockus, 2003; Olson and Olson, 2003). The outsourcing trends and the projected need for more diverse skills entail additional coverage for the SA&D course.

We discuss five reasonable but not exhaustive solutions to accommodating the additional material in two courses. These are based on our observations and experience as textbook authors as well as course recommendations in the Gorgone et al. (2006) report.

One approach to having two analysis and design courses is to have the first course focus on the SDLC, project management, defining the system vision, and defining the functional and non-functional requirements. This course can be called Systems Analysis. The second course can focus on the detailed design, implementation, testing, and support. This course can be called Systems Design. This approach provides instructors enough time to cover both topics. It is a problematic approach, however, because iterative and agile approaches do not have a clear distinction between analysis, design, and implementation. This is the approach that was once followed by the first author, but the courses were merged about a decade ago because of curriculum constraints.

Another approach is to have the first analysis and design course cover a predictive approach to the SDLC with structured analysis and design techniques. The second course can build on the first course but cover an adaptive, iterative approach with OO analysis and design techniques. Agile approaches can also be introduced. This is the approach that was followed by the second author until recently.

A third approach is to offer one course that addresses the importance of project management issues in analysis and design, and a second course that covers analysis and design techniques and models. The project management course can address basic project management principles, feasibility, information gathering, predictive versus adaptive life cycles, package selection issues, outsourcing issues, and other important aspects of IS development projects. The analysis and design techniques and models course can address business modeling, requirements modeling, analysis, design, and implementation as well as techniques based on UML diagrams. This approach is quite feasible because many curricula already have a project management course. If the program is willing to allocate a significant portion of the project management course to analysis and design methodologies and their management, the additional materials can be covered without having to introduce another course. Further, introducing a human-computer interaction course as recommended by Gorgone et al. (2006) can provide more space in the regular SA&D course, as the interface issues are transferred to the new course.

A fourth approach is based on teaching analysis and design concepts iteratively. A first course can cover the entire SDLC and introduce project management, iterative development, requirements modeling, design, and implementation. After students have completed their database and programming courses, an advanced analysis and design course can go into the same topics and techniques in more depth, reinforcing the ideas through iteration. Because it is very difficult to teach modeling without addressing both analysis and design, an iterative approach to teaching makes sense. However, the first course can be more analysis oriented while the second course can be more design and implementation oriented. Just as we learn more about the requirements by exploring design and evaluating the implementation, students learn more about the analysis and design techniques by going through a second iteration of the course in more depth. This is feasible in programs that have two courses that go in tandem with the second course geared toward implementation.

A fifth approach might be defined based on layered enterprise application architecture (Fowler, 2003). One course might introduce the three layers and cover defining the domain model and functional requirements at a high level. Then the focus can shift to designing the user interface (presentation layer) using storyboards, prototypes, and interface design principles and practices. Many IS researchers are calling for more emphasis on user interface and human-computer interaction in IS courses (Gorgone et al., 2006). With the importance of interface design and hypermedia for Web-development, there is also an appeal in

this approach for those emphasizing design for the Web. A second course could focus on detailing the functional and non-functional requirements and designing the problem domain (business logic) layer. The database course in the IS curriculum can be recast as a third course focused on designing the data source/data access layer.

Note that our recommendations do not necessarily require that we have courses such as SA&D I and SA&D II. An existing project management course or a new human-computer interaction course may provide space to cover the additional materials required to be covered in the contemporary environment.

4. REQUIREMENTS AND ANALYSIS ISSUES

If only one SA&D course is available, some difficult decisions will need to be made on removing the less useful topics. Even with two courses, too many concepts in the analysis and design courses can confuse students. Probably, the time has come to eliminate the data flow diagram (DFD). Systems development needs to incorporate component based development as part of the overall methodology. The DFD is not suitable to incorporate component development, which requires a different manner of thinking that separates the implementation of a subsystem from its interface. This separation entails an object-oriented approach. Thus, if there is room for only one course, requirements definition based on the object-oriented approach should be considered.

Most textbooks that emphasize object-oriented systems analysis and design (OOSAD) provide extensive coverage of use cases. However, there is no reason that use cases cannot be applied in other approaches. Use cases are detailed in a story-like fashion, but they capture only about one-third the total requirements (Rosenberg, 1999). Guidelines also need to be provided for capturing the remaining requirements. Should these requirements be captured as text, screen prototypes, or storyboards? What other representations would be appropriate? There is a need to provide an integrated approach to requirements capture and documentation.

The use-case approach is neither top-down nor bottom-up; it is a middle of the road approach. This can be effective in small to medium size applications. In large systems, however, a systems decomposition approach can be very effective. The idea of decomposition is central to dealing with large systems. A system must be decomposed into interrelated sub-systems recursively until a sub-system can be understood and analyzed. However, we have not seen the decomposition idea emphasized in use cases although functional decomposition has been a well accepted approach in structured methodologies. A student will not be trained to handle large systems if a SA&D course does not teach system decomposition.

Cockburn's (2001) book "Writing Effective Use Cases" provides some clue to the possibility of marrying the use case and systems decomposition ideas. He illustrates four practical levels of use cases: cloud, kite, sea level, and fish. Although his focus is mainly on the sea level and to some extent on the kite level, the different levels of abstraction of these use cases suggest that the decomposition idea is implicit even in the object-oriented world, even though Cockburn does not explicitly acknowledge it. However, multiple levels of use case abstraction can be a problem. There is no empirical study evaluating the usability of such an approach. Conversely, it might be better to have decomposition in to business areas or subsystems, with a use case always defined at the elementary business process level. This may run against the popular notion that use cases cannot coexist with systems or process decomposition, but it seems that it is a practical approach that as a start addresses an important but unresolved issue.

Lang's (2006) study shows an interesting finding – the prevalence of the entity relationship (ER) model today is as predominant as the prevalence of use cases. Given the overriding prevalence of both, it is evident that the two coexist. However, the ER model has its genesis in the data oriented approaches, while the use case diagram and use case descriptions came out of OO techniques and UML. Somehow, the two have met in practice.

It seems clear that system requirements covered in the analysis and design course can focus on systems decomposition, identifying use cases and actors, describing use cases, modeling the data requirements with a domain model class diagram or an entity relationship (ER) diagram, and documenting other requirements using a selection of text, flowcharts, UML activity diagrams, UML state machine diagrams, UML system sequence diagrams, storyboards, as preferred by the instructor. We see little lost and much gained by more narrowly defining the content by eliminating DFD modeling. There is no reason to discourage entity relationship (ER) modeling as it is very similar to domain modeling and can provide the link between the database design course and the SA&D course. In fact, the domain model class diagram used in object-oriented development is remarkably close to the entity relationship (ER) model.

5. DESIGN ISSUES

Systems design is much more complex today than it was ten years ago, making it more difficult to decide what to emphasize in the analysis and design course. First, there are Web applications that range from informational sites to dynamic data-driven sites with hypermedia functionality that includes enhanced navigation, highly visual interfaces, and multimedia content (Lang, 2006). Physical design differs when using Web technology, and applications with hypermedia functionality call for a different collection of design models and techniques.

A significant trend in today's environment is outsourcing the design and implementation given well-defined requirements are defined locally. In this case, the design part of analysis and design includes more emphasis on coordinating with external designers/developers, possibly offshore, and evaluating their work. The course might not emphasize detailed design techniques and models at all. Some courses cover a comprehensive list of design patterns used for detailed design (Fowler, 2003). Perhaps many students in IS

do not need the detailed design depth if they pursue opportunities in business modeling, requirements, and project management, while other students will need extensive design and implementation techniques.

Whether design and implementation are covered extensively depends on local needs of the program. If the business environment generally outsources design and/or implementation (as currently in the US), then certain topics (e.g., design patterns) may be of less interest. If the business environment carries on all SA&D activities (e.g., currently in Ireland), then two courses are a must, and all topics need extensive coverage. In some business environments (e.g., currently in India), design and implementation may be even more important than analysis. The local needs can, however, change rapidly in today's fast paced business environment.

For those interested in detailed design and implementation, it might be better to include detailed design and appropriate design patterns in an advanced programming/development course rather than in analysis and design. Thus, different development courses can be used to teach detailed design for each type of technology emphasized.

Another aspect of design is the need to define system interfaces to other applications as part of an integrated solution. Package solutions and components also require integration. The skills needed by an IS graduate require more about packaged solutions and integration.

If use case driven detailed OO design is taught using UML and the three-layer architecture, the UML models can become very complex and they can require deep programming knowledge. UML sequence diagrams and design class diagrams need to be studied in depth, and design patterns need to drive design decisions (Larman, 2004). It is also difficult to separate the programming and the design aspects of use case driven OO detailed design. So, it might be better to think of this as a programming/development course rather than analysis and design.

Systems analysis and design courses taught in MIS curricula focus on business applications that are usually data intensive, transaction-based applications. In the last two decades, management of data has been accomplished mainly by relational DBMS such as Oracle, DB2, Informix, SQL Server, etc. There is no Object-Oriented DBMS (OODBMS) that is even remotely comparable in sales to any of these products. Despite some promise and a fair number of products such as Gemstone, O2, Iris, ObjectStore, ORION, and Vbase, the object-oriented DBMS have achieved shallow success. Although some of the object-oriented features are slowly being incorporated in relational DBMS like Oracle, there is no evidence that these features are actually being used in business applications in a significant way. With the application end going object-oriented, but the back end (database) still predominantly relational, the end result is an "impedance mismatch" (Muller, 1999), which can create confusion in pedagogy and in practice. Muller (1999) reminds us that the use of an object-oriented approach does not mean the database disappears.

The problems with relational DBMS are well known. Yet, the tremendous productivity gains possible by using a declarative language like SQL tilt the adoption scale in favor of them. Further, the market, keen to maintain compatibility with legacy applications, is not responding with a revolutionary change to OODBMS, but with an evolutionary change to Object-Relational DBMS (ORDBMS). Overall, Object-Relational DBMS are more relational than object-oriented.

The challenge, therefore, is to develop a method that can bridge the mismatch between OOSAD and ORDBMS. The method needs to incorporate key concepts of UML such as generalization and aggregation, yet maintain the time-tested advantages of conceptual data models like the entity relationship (ER) approach, the logical data models like the relational model and its object-oriented extensions, and the implementation gains attributed to SQL. Currently, the impedance mismatch is handled by using object wrappers. An object wrapper is basically a layer on top of a conventional relational engine that simulates object-oriented features. Using object wrappers, the system appears object-oriented although the back end is relational.

6. LIFE CYCLE ISSUES

A key issue when considering what to cover in the analysis and design course is the approach taken toward the system development life cycle (SDLC). There are several options to evaluate. In a project management oriented course, there is room to discuss the predictive, waterfall SDLC and then one or more adaptive, iterative SDLCs. When time is limited, and when too many options tend to confuse students, it might be easier to focus on one SDLC model. Which SDLC to use depends on the goals of the instructor.

One SDLC model commonly used is to define the generic phases of the SDLC—Planning, Analysis, Design, Implementation, and Support. Once the activities of each phase are understood, an iterative approach to development is described using the same generic phases, but defining a series of iterations or mini projects. This model can be confusing when teaching about a waterfall model with management checkpoints and then asking students to abandon it in practice and adopt a more complicated management model for the iterative approach.

Another SDLC model is based more explicitly on the spiral model (Boehm, 1988). Many texts emphasizing object-oriented analysis and design use a spiral model to visually represent the SDLC. Planning, analysis, design, and implementation occur in each iteration, but the waterfall idea of management checkpoints after each phase and the idea of phases themselves are dropped. The advantage of a spiral model is students learn iterative development from the start. There is no need to unlearn one life cycle to understand another. We find no recent methodologies that do not advise iterative development as a best practice. However, in industry, the waterfall model is alive and well (Lang, 2006) and students might benefit from understanding it. Also, the spiral model representation mitigates project planning and

project management issues that need to be addressed in analysis and design courses.

An attractive alternative to the generic SDLC and the spiral model is provided by the Unified Process (UP) life cycle model (see Figure 1). The UP life cycle retains project phases and management checkpoints based on phases. But the phases are not named planning, analysis, design, and implementation. Therefore, project management issues in a sequential life cycle can be discussed and modeled. The UP phases are Inception, Elaboration, Construction, and Transition (Kroll and Kruchten, 2003). Each phase consists of one or more iterations. Once these concepts are learned, there is nothing to unlearn. However, it is a more complicated model of systems development project management.

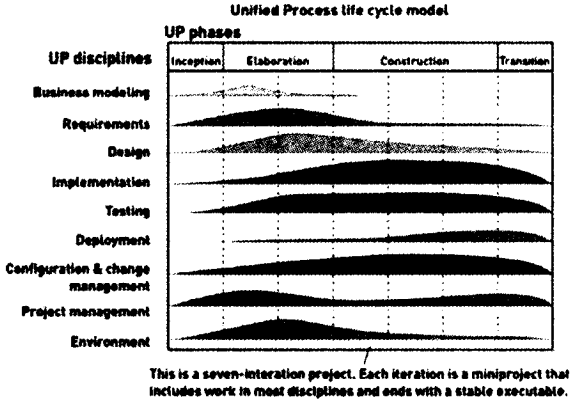


Figure 1: The Unified Process Life Cycle Model (reprinted courtesy of Thompson Course Technology)

Instead of parsing the field into analysis and design, the UP defines nine disciplines used by system developers. A developer draws on each of these disciplines during any given iteration. The disciplines are business modeling, requirements, design, implementation, testing, deployment, configuration and change management, project management, and managing the development environment. The concepts and techniques of each discipline can be discussed in any order and in any iteration. Discussing the nine disciplines gives students a useful framework for understanding the broad set of knowledge and skills required in the IS field.

7. COMPLEXITY OF UML

The advent and popularity of Unified Modeling Language (UML) is impacting the Systems Analysis and Design (SA&D) course. UML, which is based on object-oriented concepts, is now taught in practically all MIS programs. Even textbooks that have structured analysis and design as the main theme devote at least a chapter or two to UML. But in recent empirical studies (Siau and Cao, 2001; Siau, Erickson, and Lee, 2005; Dobing and Parsons, 2006), UML has been found to be complex by a number of analysts. UML has too many diagrams and the usefulness of some of the diagrams has been questioned. The use case diagram, the class diagram, and the sequence diagram have been found to be the most useful. Lang's (2006) study shows that the activity diagram is pervasive, and the state machine diagram has fair usage. The communication diagram (which was earlier called collaboration diagram) has been found to be redundant. For the student learning UML for the first time, the scope needs to be restricted to a small subset of the diagrams. An instructor needs to customize the coverage of UML based on prior skill levels of the students.

In the Siau and Loo (2006) study, students found that the transition from the structured concepts to UML was difficult. Prior programming knowledge helped, but those who had procedural backgrounds had more problems than those who had object-oriented backgrounds. As mentioned earlier, UML needs to be covered in a SA&D course that employs object-orientation. Further, students were found to have difficulty in deciding between the activity diagram and the state machine diagram. This is a natural consequence of having too many models. They found it difficult to house constraints. Although there are languages like Object Constraint Language (OCL), which are specifically designed to model constraints, one wonders the usability impact of adding more mechanisms for modeling. Also in the Siau and Loo (2006) study, students reported having trouble understanding the relationships between the diagrams and putting them together to get an overall picture. These issues suggest that the systems analysis and design area today has a wider scope, and that compressing a large number of topics into one course is problematic. Trying to squeeze a vast number of topics into one course, putting a quick fix on it hoping the novice student mind will somehow absorb the material, is an approach that is unlikely to work in practice.

8. RECOMMENDATIONS

The substantial recent developments in systems analysis and design methodologies models, tools, and techniques clearly call for a careful and detailed evaluation of the new and expanded role of SA&D in the curriculum. In this paper, we have outlined the major issues affecting analysis and design that call for a significant revision of SA&D courses. Each IS educational program will have its own traditions and needs that will require specific solutions. However, each program should consider how to incorporate the three fundamental changes the SA&D curriculum – the diversity of methodologies, the emergence of the iterative approach, and the increasing adoption of the agile approach – as discussed in this paper.

There is no one comprehensive methodology that can be recommended for adoption by the IS educational community. Research shows there is great diversity in methodologies used in practice. Although best practices and development principles seem to be moving in an adaptive and agile direction, there are many life cycle models, techniques, models, and tools that can be combined into reasonable hybrid methodologies. We recommend that at some point in the SA&D curriculum, specific methodologies such as the Rational Unified Process (RUP), Extreme Programming (XP), Adaptive Software Development, Crystal, and Scrum might be surveyed, but mainly to show the similarities in terms of best practices and development principles.

Given the number of issues in analysis and design, we recommend serious consideration of including two courses in the IS curriculum that can cover the range of topics on analysis and design techniques as well as system development methodologies. We outlined five possible pairs of courses to consider, and listed the key topics. In the two courses the following topics should be covered: project management, use cases, UML modeling, iterative development, agile development, package selection and integration, Web development, outsourcing/off shoring, and at least conceptual design for system controls, user interfaces, and database management. Eventually, local needs and preference must play a role in the decision.

In terms of functional requirements and analysis issues, we argue for employing a use case driven approach. We feel it is time to drop the data flow diagram (DFD) model as it does not support component development, a common practice today. Use cases are also more prevalent than DFD's in practice. Non-functional requirements can be modeled in a variety of ways, and procedural logic can be taught using UML activity diagrams or traditional flow charts. Because use cases are not inherently object-oriented, a use case driven approach can be used with a more traditional approach to technology. Employing use cases does not necessarily require switching to OO. While we suggest that the time has come to drop the data flow diagram (DFD), the entity relationship (ER) diagram and diagramming technique are widely used and quite appropriate for data requirements and database design.

We recommend that the SA&D courses use UML, the long-needed standard, whenever possible for modeling. We recommend that whatever approach to SA&D is emphasized, it is probably best to focus on a few key models and techniques rather than feeling the need to cover everything as a rapid survey. Therefore, although we recommend using UML, we do not argue for covering every UML diagram. The use case diagram, class diagram, and sequence diagram provide the best overall view. The instructor needs to also demonstrate how the diagrams are related to each other, and how they present an integrated picture.

We propose that for many students in western countries, detailed system design techniques and models may not be as important as methodology knowledge, requirements models, and project management skills. First, outsourcing and off-shoring trends point to the need for increased expertise in analysis and project management skills and less on detailed design and programming. Similarly, component-based development, service oriented architecture (SOA), and package integration call for increased analysis and project management skills. Finally, specialized technical environments and development tools call for more specialized detail design techniques that might be better taught in an advanced programming/development course rather than in the analysis and design sequence. Thus, students more interested in detailed design and programming will be exposed to detailed design in the right course.

Although there are many ways to teach and model an adaptive SDLC, we also argue that teaching one life cycle model might be appropriate for initial learning. The Unified Process (UP) life cycle is a good choice because it models phases, iterations, and activities in all disciplines throughout the project. The UP life cycle and disciplines can be taught without having to adopt other details of the Rational Unified Process.

9. CONCLUSIONS

There are many issues related to SA&D that require research, discussion, and debate. The paper outlined and discussed many of the issues and made some recommendations. Often, the devil is in the details, and it is likely that the SA&D course will remain challenging, controversial, and important. We feel that in the next few years, a generally accepted approach to the SA&D curriculum will emerge. In the meantime, everyone involved in teaching the SA&D curriculum should be open to trying some new approaches.

10. REFERENCES

Ambler, S. W. and Jeffries, R. *Agile Modeling: Effective Practices for Extreme Programming and the Unified Process*, Wiley, 2002.

Auer, K. and Miller, R. *Extreme Programming Applied*, Indianapolis, IN: Pearson Education, 2002.

Beck, K. *Extreme Programming Explained,* Reading, MA: Addison-Wesley, 1999.

Boehm, B. "A Spiral Model for Software Development and Enhancement," Computer, Vol. 21, 1988, pp. 61-72.

Boehm, B. and Turner, R. *Balancing Agility and Discipline*, Boston, MA: Addison Wesley, 2003.

Booch, G., Rumbaugh, J., and Jacobson. I. The *Unified Modeling Language User Guide*, NJ: Addison Wesley, 1999.

Brown, D. and Wilson, S. *The Black Book of Outsourcing: How to Manage the Changes, Challenges, and Opportunities*, Wiley, 2005.

Chen, P.P. "The Entity-Relationship Model - Toward a Unified View of Data," *ACM Transactions on Database Systems*, 1(1), 1976, pp. 9-36.

Coar, K. "The Sun Never Sits on Distributed Development," *ACM Queue*, vol. 1, no. 9, pp. 32-39, 2003.

Cockburn, A. *Writing Effective Use Cases*, NJ: Addison Wesley, 2001.

Cockburn, A. *Crystal Clear: A Human-Powered Methodology for Small Teams*, Agile Software Development Series, 2004.

Dobing, B. and Parsons, J. "How the UML is used," *Communications of the ACM* (forthcoming).

Fowler, M. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*, Third Edition, Addison-Wesley Professional, 2003.

Gorgone, J. T., Gordon B. Davis, Joseph S. Valacich, Heikki Topi, David L. Feinstein, and Herbert E. Longnecker. "IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems," *The Data Base for Advances in Information Systems*, Volume 34, Number 1, Winter 2002, pp. 1-52.

Gorgone, J. T., P. Gray, T. Stohr, Joseph S. Valacich, and R.T. Wigand. "MSIS 2006: Model Curriculum and Guidelines for Graduate Degree Program in Information Systems," Communications of the AIS, Volume 17, 2006, pp. 1-56.

Herbsleb, J.D. and Mockus, A. "An Empirical Study of Speed and Communication in Globally Distributed Software Development," *IEEE Transactions on Software Engineering*, Vol. 29, No. 6, 2003, pp. 481-494.

Highsmith, J. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. Dorset House, 2000.

Jacobson, I. *Unified Software Development Process*, Cambridge University Press, 2000.

Kroll, P. and Kruchten, P. *The Rational Unified Process Made Easy: A Practitioner's Guide to Rational Unified Process,* Addison-Wesley Professional, 2003.

Lang, M. "New Branches, Old Roots: A Study of Methods and Techniques in Web / Hypermedia Systems Design," *Information Systems Management* (forthcoming), 2006.

Larman, C. *Agile & Iterative Development: A Manager's Guide*, Boston, MA: Pearson Education, 2003.

Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd Edition), Prentice Hall PTR, 2004.

Muller, R.J. 1999 *Database Design for Smarties*, CA: Morgan Kaufmann, 1999.

Olson, J.S. and Olson, G.M. "Culture Surprises in Remote Software Development Teams," *ACM Queue*, vol. 1, no. 9, pp. 52-59, 2003.

Rosenberg, D. *Use Case Driven Object Modeling*, MA: Addison Wesley, 1999.

Sahay, S. "Global Software Alliances: the Challenge of Standardization," *Scandinavian Journal of Information Systems,* Vol. 15, 2003, pp. 3-21.

Schwaber, K., and Beedle, M. *Agile Software Development with Scrum*, Upper Saddle River, NJ: Prentice-Hall, 2002.

Siau, K. and Cao, Q. "Unified Modeling Language (UML) – A Complexity Analysis," *Journal of Database Management,* 12(1), 2001, pp. 26-34.

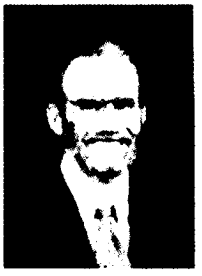
Siau, K., Erickson, J., and Lee, L.Y. "Theoretical vs. Practical Complexity: The Case of UML," *Journal of Database Management,* 16(3), 2005, pp. 40-57.

Siau, K and Loo, P. "Identifying the Learning Difficulties with Unified Modeling Language (UML)," *Information Systems Management* (forthcoming), 2006.

Teorey, T.J., Yang, D., and Fry, J.F. "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Mode," *Computing Surveys*, 18(2), 1986, pp. 197-222.

AUTHOR BIOGRAPHIES

Dinesh Batra is Professor of MIS in the College of Business Administration at the Florida International University. He is a co-author of the book *Object-Oriented Systems Analysis and Design* published by Pearson Prentice-Hall. His publications have appeared in, *Communications of the ACM Management Science, Journal of MIS, Data Base, European Journal of Information Systems, International Journal of Human Computer Studies, Computers and Operations Research, Information and Management, Journal of Database Management, CAIS, Decision Support Systems,* and others. He has served as the President of the AIS SIG on Systems Analysis and Design.



John W. Satzinger is Professor of Computer Information Systems at Missouri State University. He is co-author of five system development book titles published by Thomson Course Technology, including *Systems Analysis and Design in a Changing World, 4th Edition* (2007) and *Object-Oriented Analysis and Design with the Unified Process* (2005). He has also published numerous articles about analysis and design and human-computer interaction in journals such as *Information Systems Research, Journal of MIS, Database*, and *Communications of the AIS*. He received his Ph.D. in MIS from the Claremont Graduate University in 1991.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096