Communications of the Association for Information Systems

Volume 20 Article 31

October 2007

Analysis and Design in the IS Curriculum: Taking it to the Next Level

John W. Satzinger
Southwest Missouri State University, jws086f@smsu.edu

Dinesh Batra
Florida International University, batra@fiu.edu

Heikki Topi
Bentley College, htopi@bentley.edu

Communications of the Association for Information Systems

ANALYSIS AND DESIGN IN THE IS CURRICULUM: TAKING IT TO THE NEXT LEVEL

John W. Satzinger
Missouri State University

Dinesh Batra
Florida International University
DineshBatra@business.fiu.edu

Heikki Topi
Bentley College

ABSTRACT

Recent surveys of methodologies and techniques currently used in organizations for developing information systems indicate 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. Several course-related and environment governed trends seem to impact the coverage, including the growing popularity of object-oriented techniques, the shortening of the life cycle and the emergence of the iterative approach, the increasing adoption of the agile approach, the rising importance of UML, the outsourcing trend leading to global distribution of SA&D work, and the rate of change in the technical and business environments. The scope of the SA&D course has increased. Yet, most MIS degree programs have just one SA&D course. 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. A panel at the Americas Conference on Information Systems (AMCIS) 2007 conference evaluated how the trends impact the coverage of the SA&D course and made recommendations on how these trends can be addressed. Based on the panel discussion, this paper tackles the many challenges of teaching analysis and design in the IS curriculum and taking it to the next level.

Keywords: Systems analysis, systems design, systems methodologies, systems techniques, IS teaching

I. INTRODUCTION

The majority of system failures can be attributed to problems that arise during the systems analysis and design phases [Iivari, Parsons, and Hevner, 2005]. One of the most significant changes to the information systems (IS) field in recent years has involved system development methodologies and system development practices [Bajaj et al, 2005]. Object-oriented techniques and technologies have made significant inroads into business systems development [Johnson, 2002; Mahapatra, Nerur, and Slinkman, 2005], which has required rethinking many aspects of analysis and design. Additionally, methodologies based on the use of the Unified Modeling Language (UML) have emerged as dominant modeling approaches for analysis and design now that UML has been accepted as a standard by the Object Management Group (OMG) [Kobryn, 1999; Batra, 2008]. Another important trend has been toward the use of adaptive, iterative life

cycle models to deal with the complexity of systems development, in place of the predictive waterfall model [Larman, 2005]. These changes have required rethinking much of what has been traditionally taught in systems analysis and design (SA&D) courses in the IS curriculum.

Other trends affecting system development involve project management, technology, and sourcing options that affect how and where key SA&D concepts and techniques are taught. For example, agile development as a general approach [Larman, 2004] and specific agile development methodologies, such as eXtreme Programming [Beck, 2000], Scrum [Schwaber and Beedle, 2002], and Crystal [Cockburn, 2005], present challenges and opportunities when deciding how to manage a development project in SA&D. Web technologies and service oriented architecture (SOA) greatly impact how system design is approached in SA&D courses, including designing security and controls. For many years now, SA&D instructors have struggled to deal with addressing issues related to software package selection and implementation, including enterprise resource planning (ERP) solutions [Ragowsky, Somers, and Adams, 2005]. At the same time, the variety of sourcing models for systems development have complicated managing the development process and supporting the project team [Cullen and Willcocks, 2003]. Teams and team members are increasingly geographically distributed, multicultural, and multi-lingual [Olson and Olson, 2003]. At the same time, the rate of change in the business and technical environments continues to increase [Reeves, 1999], adding even more pressure to SA&D instructors to be sure they are teaching up to date and effective tools and techniques.

A panel session at the Americas Conference on Information Systems (AMCIS) 2007 conference at Keystone, Colorado, was motivated by the need to examine the systems analysis and design component in the IS curriculum to discuss what methodologies, techniques, models, and tools need to be taught in the IS Systems Analysis and Design course. A number of issues seemed pertinent. 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 data flow diagram (DFD's)? Should use cases be the basic approach to the specification of functional requirements? 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 or is the web yet another implementation platform? Given the outsourcing environment, how much design and implementation should be covered as compared to requirements specification, analysis, and project management? What is the role of business process modeling and design in SA&D? Can SA&D be taught effectively without considering the details of technical design? Should an IS program consider a second course in SA&D to fit the topics? Or, should the SA&D instructor coordinate with the instructor who teaches the project management (PM) course, which can incorporate the management principles of the adaptive versus predictive and the disciplined versus agile approaches to development?

The panel attempted to address such questions. It was attended by about 50 people, who actively participated in the discussion. The interest in such systems analysis and design topics is corroborated by trends such as a steady increase in membership and activities by Association for Information Systems Special Interest Group on Systems Analysis & Design (AIS SIGSAND), an active European SIGSAND chapter, and interest in other regions in the world to open similar chapters.

This paper surveys the topics addressed at the AMCIS 2007 panel. First, recent research is reviewed that demonstrates the complex and diverse state of practice that makes it difficult to define the current requirements for an effective SA&D course. The effect of past and proposed curriculum models on the SA&D course is then discussed. Next, in order to discuss the SA&D course systematically, four fundamental changes affecting analysis and design and the SA&D course are identified and discussed. Specific questions and issues under each fundamental change are raised. Additionally, there are some controversial issues in academe that potentially

affect the ability of IS researchers to properly teach the SA&D course. Finally, we make specific recommendations for improving teaching practices in SA&D.

II. STATE OF PRACTICE

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.

Table 1. Use of various methodologies in practice (Lang, 2006)

METHODOLOGY	**ADOPTION**
Hybrid, customized, in-house	23%
Traditional “legacy”, Waterfall	22%
Agile development methods	15%
Tools based (*e.g. PHP, J2EE, InterDev*)	14%
Incremental (*e.g., RUP, Spiral*)	13%
Object-oriented (*e.g. OOAD, UML)*	8%
Ad Hoc	8%
Human Factors Engineering methods	5%
Technique-driven (e.g., *Storyboarding)*	6%
Specialized for Web/hypermedia	5%

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). Dobing and Parsons [2008] show similar practices with class, use case, and sequence as the most used among the UML diagrams. It appears likely that the use 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 from legacy techniques (e.g., entity relationship and flowcharts).

III. CONSTRAINTS OF IS CURRICULUM MODELS

Some of the key factors affecting the SA&D course are the IS curriculum models at undergraduate and graduate levels [Gorgone et al., 2002; Gorgone et al., 2006]. IS 2002, the undergraduate model curriculum, is highly prescriptive in nature and offers little choice for schools adopting it. The curriculum consists of 10 required courses and no electives. SA&D has, however, a significant role within this model curriculum: various aspects of systems analysis and design are covered in four courses included in the curriculum, and the IS 2002 document [Gorgone et al.,

2002] goes as far as stating that Information Systems is equal to Technology-enabled Business Development (including Systems Analysis and Design, Business Process Design, Systems Implementation, and IS Project Management).

Table 2. Use of various models and techniques in practice [Lang, 2006]

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

IS 2002.7 Analysis and Logical Design is a traditional SA&D course that focuses on the early stages of the systems development life cycle. In addition, IS 2002 includes two "Physical Design and Implementation" courses (IS 2002.8 Physical Design and Implementation with DBMS and IS 2002.9 Physical Design and Implementation in Emerging Environments) and the capstone IS 2002.10 on Project Management and Practice. IS 2002.8, .9, and .10 all include elements that are within the broad umbrella of SA&D, particularly if we take the "D[esign]" in SA&D seriously. Some of the topics in IS 2002.8 include "structured and object design approaches," "design tools," and "database implementation including user interfaces and reports," all topics associated with SA&D. In 2002.9, relevant topics are "structured, event-driven, and object-oriented application design," "testing," "software quality assurance," and "multi-tiered architectures and client independent design." Finally, the topics in IS 2002.10 include "managing the system life cycle," "requirements determination," "design," "project tracking, metrics, and system performance evaluation." Clearly, IS 2002 acknowledges that SA&D is in the very core of the Information Systems profession and has characteristics that demonstrate this in a very concrete way.

However, the trends listed at the beginning of this article suggest that IS 2002 needs to be revised. The key ideas of the curriculum were developed in the mid-1990s, and thus, the time elapsed since alone is a strong enough reason to review the curriculum. Both the technical and the business environments have changed dramatically since the core architecture of IS 2002 was developed, including fundamental changes such as the ubiquitous connectivity offered by the Internet, the emergence of the Web as a platform for communication and applications, the highly widespread use of ERP systems as the foundation for organizational information systems solutions, and the transformation of the nature of IS/IT work to a fully global and distributed model. The trends provide an opportunity for the Information Systems community to rethink and potentially redefine its identity – at least for the external constituencies, the field is defined at least as much through what we teach as it is through our research output. A new model curriculum is clearly needed.

A joint Association for Computing Machinery (ACM)/AIS project to revise the Information Systems undergraduate model curriculum was launched in early 2007, and the first round of results of the committee's work were discussed and made available to the public at the AMCIS 2007 meeting [Topi et al, 2007]. One of the key changes included in the current proposal is the separation of the core of the curriculum from electives, with the recognition that the core concepts and skills in

information systems may be integrated in a meaningful way with concepts and skills in a number of technology specialties and domain areas. In addition, the proposal acknowledges that even the core concepts and skills are not covered at the same level of detail in all programs.

From the perspective of the current discussion, a key question, of course, is the role of SA&D in the revision. The positive news is that SA&D is included in the proposed core together with Foundations of Information Systems, Data & Information, IT Infrastructure, Project Management, and Application Development (see Table 3, also available at http://blogsandwikis.bentley.edu/iscurriculum/index.php/New_Curriculum_Structure_and_Content). Note that the dark circles in the table represent significant coverage, while the light circles represent some coverage. There have been virtually no critical voices suggesting that SA&D should not be part of the core. Because of the change in the nature of the curriculum, the extent to which SA&D will be covered is not prescribed at the same level of detail as it was in the previous curriculum. At the same time, a school that wants to develop a program that has a very strong emphasis on SA&D can do that and still stay within the boundaries of the model curriculum. It is unlikely that the model curriculum will specify at a detailed level a specific approach to teaching SA&D in terms of either pedagogy or content. For example, it is likely to stay indifferent regarding the selection between object-oriented or structured approaches to SA&D or the choice between different modeling grammars.

Table 3: SA&D within the proposed IS curriculum draft representation Fall 2007.

Structure of the IS Model Curriculum [edit]

Career Track:	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
Core Topics:																		B = Business Intelligence Manager
Foundations and Role of IS	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	C = Business Process Analyst
Data & Information	●	●	○	●	●	○	○	●	●	○	●	○	○	○	○	○	○	D = Database Administrator
Systems Analysis & Design	●	○	●	○	○	○	○	○	○	○	○	○	○	○	●	●	●	E = Database Analyst
IT Infrastructure	○	○	○	●	○	●	○	○	●	●	○	○	●	●	○	○	○	F = e-Business Manager
Project Management	●	○	○	○	○	●	○	○	○	○	●	○	○	○	●	●	●	G = ERP Specialist
Application Development	●	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	H = Information Auditing and Compliance Specialist
																		I = IT Architect
Elective Topics:																		J = IT Asset Manager
Business Process Management			●			○	○	○		○	●				○			K = IT Consultant
Collaborative Computing						○								○			○	L = IT Operations Manager
Data Mining / Business Intelligence		●		●	●	○	○	○	●		○	○	○	○	○		○	M = IT Security and Risk Manager
Enterprise Architecture				○	○	○	○	○	●	○	○		○	○	○		○	N = Network Administrator
Enterprise Systems			●	○	○	○	●	●	○		●	●	○	○				O = Project Manager
Human-Computer Interaction	●	○					○					○				●		P = User Interface Designer
Information Search and Retrieval		○		○	●								○				●	Q = Web Content Manager
IT Audit and Controls	○		●	○	○	○	○	●		●	○		○	○	○		○	
IT Security and Risk Management	○			○	○	○	○	●	●	○	○		●	●	○		○	
IS Management and Strategy	○		○			○	○		●		●				○			
Knowledge Management		●		○		○	○			○								
Social Informatics													○		○			

IV. FOUR FUNDAMENTAL CHANGES AFFECTING TEACHING SYSTEMS ANALYSIS AND DESIGN

Given the state of current practice and constraints on the IS curriculum, four fundamental changes affecting the practice and teaching of SA&D were proposed by the panel as a framework for discussing the SA&D course. Although there is some overlap and some additional issues that might fall outside of this framework, most of the key questions listed earlier fall within the framework. The four fundamental changes include:

- The Change from Structured to Object Oriented Models/Techniques

- The Change from Waterfall (Predictive) to Iterative (Adaptive) Life Cycles
- The Change from Formal, Disciplined Process to a more Agile Process
- The Change from Internal SA&D Processes to Globally Distributed, Outsourced Processes.

THE CHANGE FROM STRUCTURED TO OBJECT ORIENTED MODELS/ TECHNIQUES

Although not specifically object-oriented, use cases are often identified with SA&D courses that cover UML and the object-oriented approach to analysis and design [Dobing and Parsons, 2008]. Covering use cases is often done at the expense of the time used for data flow diagrams (DFD's) and the data dictionary. The panelists proposed that use cases are effective for capturing functional requirements, although other requirements such as usability, reliability, performance, and security (the URPS in FURPS; see Larman, 2005) also need to be documented. A key question raised by the audience was whether functional decomposition can be taught effectively with use cases. Data flow diagrams provide a clear hierarchical decomposition that, even if not used in practice, can be an effective approach to teaching decomposition. Another use case topic is the emphasis on use case diagrams versus use case descriptions. Teaching use case diagrams without use case descriptions provides little benefit, in the same way as using use case diagrams without the descriptions provides few benefits in the systems development process. Use case descriptions, written using one of the popular templates (e.g., Cockburn [2001]), document important information regarding the interaction between the system and its users that can be used throughout development but is not provided by DFD's and supporting documentation. In a use case driven project, use cases provide the foundation for functional requirements and, through those, for user interface design, for implementation, for testing, for training, and for deployment. The panelists argued for eliminating DFDs in favor of use cases and UML activity diagrams for both traditional structured development and for OO development, although several members of the audience found DFD's still useful and important. It was commonly accepted that use cases are not in any specific way restricted to object-oriented analysis and design, even though their widespread use originated from the communities advocating object-oriented SA&D.

The emergence of the Unified Modeling Language [Kobryn 1999, 2004] as the dominant grammar for modeling diagrams also raises a series of questions about teaching SA&D. The panelists proposed that instructors do not have much choice but to cover UML: it has become an essential component of any IS professional's toolset. A widely recognized problem is the complexity of UML [Siau, Erickson, and Lee, 2005; Siau and Loo, 2006]. The specification includes thirteen diagrams each with many options and potential levels of detail. Surveys on practice have generally concluded that the class diagram, the use case diagram (together with the narrative description), and the sequence diagram can serve as key system modeling artifacts. Not every variation and detail of each model is necessary. The key point is to show how models fit together and how models evolve from requirements to design to implementation while maintaining the conceptual consistency.

One key issue is the use of entity relationship diagrams (ERD) and domain model class diagrams to model the same system concepts. The panelists agreed these diagrams can be used interchangeably, and because of the importance of the ERD to database analysts, today's SA&D student should be familiar with both. Students should also learn that the same diagram grammar can be used for multiple purposes: the UML class diagram can be used for representing both a high-level domain model and a very detailed system design description, or the UML sequence diagram can be used for modeling both detailed user-system interaction and the communication sequences between the internal objects. UML can be used at various levels of detail and abstraction. In his popular book, Martin Fowler [2004] identified three possible modes of applying UML in software development: sketch, blueprint, and programming language.

The audience was asked whether they had one or two courses available for covering SA&D in their undergraduate programs. Many were limited to one course. Some with two courses covered

a traditional approach in the first course and an OO approach in the second course. Some were forced to cover elements of both in one course. A few covered just OO in the one available course, and some covered just traditional SA&D in the one available course. Several covered introductory and then advanced OO SA&D over two courses. A key question to ask is whether it is desirable to cover both traditional and OO approaches in a curriculum. The panel agreed that in the recent past, covering both in some mix was desirable, with more and more emphasis on OO over time. But now it was clear that a use case driven approach coupled with UML and OO design was the best option. Audience comments addressed local realities faced when company recruiters wanted more traditional background. Others commented that the job market overall expected the latest models and techniques. Perhaps teaching some historical context describing traditional models and techniques might be enough for many instructors and programs.

THE CHANGE FROM WATERFALL (PREDICTIVE) TO ITERATIVE (ADAPTIVE) LIFE CYCLES

Another key change is the move from using a predictive, waterfall approach to the SDLC to the newer adaptive, iterative approach to the SDLC. The panelists agreed that the question of which approach to use is dependent on the nature of the project. The predictive, waterfall SDLC has always been effective for projects that have well-defined requirements, well-understood technology, and low overall risk. Scope, requirements, technology, required tasks, and staffing levels are predictable enough to allow the project to be planned out in advance [Boehm and Turner, 2003]. Many IS system development projects are still planned and managed this way. Often team members plan and report the project as though it is waterfall, but in reality they define work tasks and deliverables with overlap and iteration. However, many (perhaps most) development projects today have vague and changing requirements, new less stable technology, and diverse staff members with varying experience. In addition, most current projects are built on the top of existing application and technology infrastructure currently in use, often utilizing enterprise systems or other packaged systems as part of the systems architecture. These projects require more flexibility in planning and management. Iterative SDLCs provide for embracing change and adapting to change once the project is underway.

As with traditional versus OO, it is not clear the current SA&D course can cover both predictive and adaptive in one course. If that is the case, which one should be emphasized? For teaching, it makes sense to cover planning concepts, analysis/requirements concepts, design concepts, and implementation concepts sequentially while simultaneously identifying them as conceptually separate activities that can and often are performed iteratively. Audience members suggested it might be better to start with the predictive approach and then also discuss the reality of iteration. Others suggested that teaching an adaptive methodology from the start is a better solution, including the life cycles from the Unified Process (UP) [Bergstrom and Raberg, 2004; Satzinger and Jackson, 2003] and even agile development such as eXtreme Programming (XP) [Beck, 2000], and Scrum [Schwaber and Beedle, 2002]. The panelists raised the question of whether teaching one life cycle for large projects (perhaps the traditional SDLC or the iterative UP) and a different life cycle for small projects was a good approach. Again, those with only one available SA&D course saw no clear solution. Finally, the panelists suggested a project management course, separate from the SA&D course, might cover the predictive approach generally and as it applies to some system development projects. Additionally, package selection and ERP projects might fit into a project management course and be focused on predictive SDLCs.

The panel proposed that the Unified Process (UP) life cycle model might be a good model to use when teaching SA&D because it can reduce complexity by separating the life cycle phases from the development and support disciplines. The UP Phases shown in Figure 1—inception, elaboration, construction, and transition—can be used to explain project planning and management checkpoints. The UP Disciplines can be used to teach the models, tools, and techniques used throughout the development project. Separation of the two helps students understand the disciplines as activities that can be combined in a number of different ways into development processes. Once the iterative UP life cycle is understood, the disciplines can be

discussed in depth sequentially even though they are typically used iteratively. Several audience members objected to the UP life cycle as being overly complex and difficult to explain, arguing against the panelist's position. Perhaps much like OO concepts and techniques were confusing and complex to experienced developers years ago, the UP life cycle and separate disciplines are confusing and complex to experienced developers and many IS instructors at this point. The paradox of apparent complexity for a framework thought to reduce complexity reveals some important researchable issues.

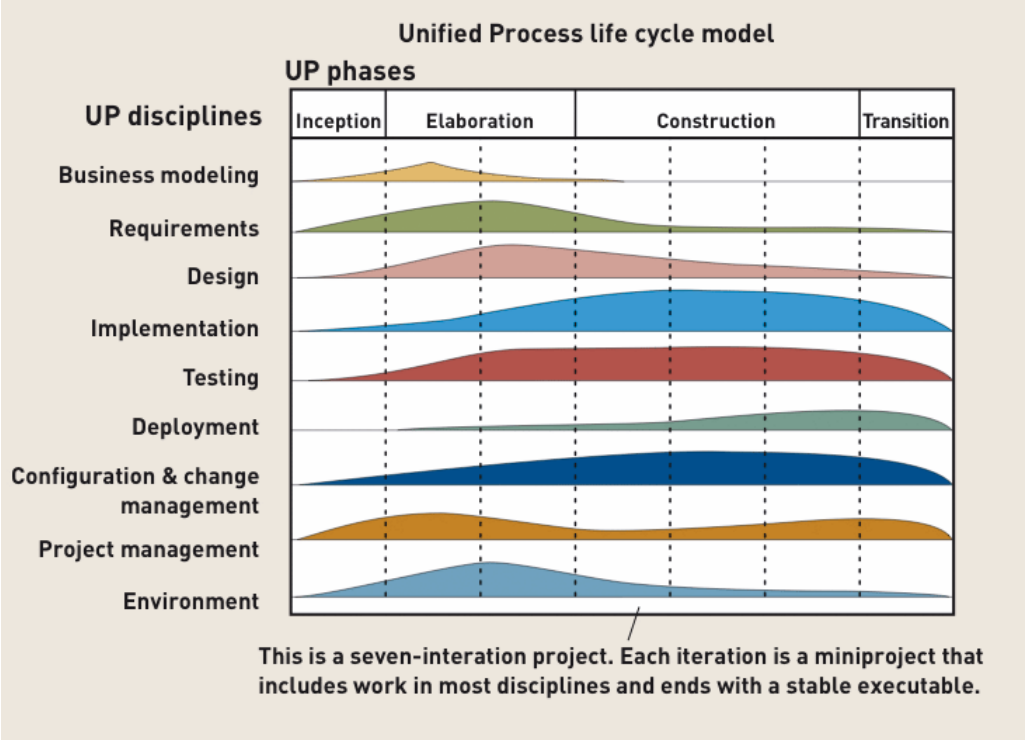

Figure 1. The Unified Process Life Cycle Model (courtesy of Thompson Course Technology)

The panelists concluded that the UP life cycle can be used in a predictive project or an adaptive project, even though iterations are explicitly included in the model. As such, one project can be planned out more completely in advance, even if the primary use of the UP model is for adaptive projects. The panelists also proposed that the UP life cycle can be used in a project that does not follow the details of the Unified Process methodology.

THE CHANGE FROM FORMAL, DISCIPLINED PROCESS TO A MORE AGILE PROCESS

Related to the question about the choice between predictive (waterfall) and adaptive (iterative) processes is the question about the impact of the recent popularity of agile development methods, such as eXtreme Programming (XP) [Beck, 2000] and Scrum [Schwaber and Beedle, 2002]. These and similar methodologies, while different from each other, have many characteristics in common: brief iterations with frequent releases, focus on functioning code, developing tests before code, de-emphasizing documentation as a mechanism for communication within the project, close cooperation between the project team and its customers, and strong emphasis on continuous customer feedback.

The panel discussed the impact of applying agile principles within SA&D courses and, specifically, for SA&D project work. In a truly agile approach, many of the cornerstones of traditional SA&D courses are either put aside or significantly de-emphasized. The SDLC concept is very different in the agile environment, which has a large number of small iterations and strong emphasis on developing functional code. Many developers following agile practices do not use a

stage or phase concept for organizing their work, and they would detest using the term SDLC to describe their work. Therefore, integrating traditional stage-based SDLC principles with a genuine agile approach can be challenging. Whatever choices are made regarding the role of agile development, it is important that the different approaches are not mixed in a way that prevents the students from understanding their true nature.

The agile methods philosophically downplay the importance of comprehensive and detailed models. This creates an interesting challenge for those faculty members for whom modeling is in the very center of not only SA&D but the entire field of Information Systems. The Agile Manifesto (http://agilemanifesto.org/principles.html) states that "Simplicity – the art of maximizing the amount of work not done – is essential" and "Working software is the primary measure of progress." These two principles combined would lead one to believe that agile approaches include no modeling work at all. This is not, however, the case; instead, in the agile world, models are often used in the "sketch" mode, are not perfected beyond their use as a communication tool, and are, in many cases, not updated unless the update is necessary for a specific purpose. Moving to an agile approach in an SA&D course would not take away the need for covering modeling techniques and languages (such as UML), but their centrality in the development process could potentially be reduced significantly. Scott Ambler states this principle as follows: "…a good rule of thumb to ensure that you're traveling light [one of his Agile Modeling principles] is that you shouldn't create a model or document until you actually need it" (http://www.agilemodeling.com/essays/agileModelingXP.htm).

There are, however, pedagogically interesting approaches that integrate agile principles with the fundamental approach underlying the Unified Process. The panelists have found Craig Larman's Agile UP [Larman, 2005] an interesting and pedagogically useful approach that maintains the integration between UML artifacts and the conceptual integrity of the models throughout the project lifecycle while emphasizing the importance of keeping the project agile and light. Larman directly recommends that we should "model and apply the UML for the smaller percentage of unusual, difficult, tricky parts of the design space" instead of applying it to all of software design. Based on our experience, Larman's approach may be one of the better ways to include both UP and agile thinking in the same course. Figure 2 includes a schematic representation of how a limited set of UML artifacts can be effectively integrated in Larman's version of Agile UP.

The panel did not address the question whether or not a single SA&D course could successfully be built around agile principles using a pure agile approach. It appears that while it might be possible to do, it is probably not pedagogically wise because for most students, it is important that they gain skills in developing integrated models using a widely used modeling grammar (such as UML). Using an approach that combines agile principles with a subset of UP (such as Larman's Agile UP) might, however, offer an interesting opportunity to bring together different perspectives.

THE CHANGE FROM INTERNAL SA&D PROCESSES TO GLOBALLY DISTRIBUTED, OUTSOURCED PROCESSES

Since the discussion on the first three trends took most of the time allocated to the panel session, the panelists were not able to get into any depth on the remaining issue of distributed development and outsourcing. It seemed that the attendees have not even started thinking about how to address this trend. So, we present our opinion of the issues involved, and whether these can be covered in an SA&D course.

Suppose a US-based IS company embarks on a systems development project and decides to outsource the coding and testing aspects to a company in India. Given that there will be temporal, geographical, language, and cultural differences between the two countries, it is evident that project management can become more complex. What kinds of problems do cultural differences create and how can we resolve them? Given that the time zone difference between the client and the vendor site is 10-12 hours, what are the requirements for maintaining effective communication? Are all sites following the same methodology, or are their different mindsets? How can changes requested by customers be transmitted by the outsourcing client to the

vendors? How can common problems with the accuracy of specifications be avoided in an environment in which written specifications are often the primary method of communication? These issues raise the question: should the project management issues of outsourced projects be included in the systems analysis and design course?

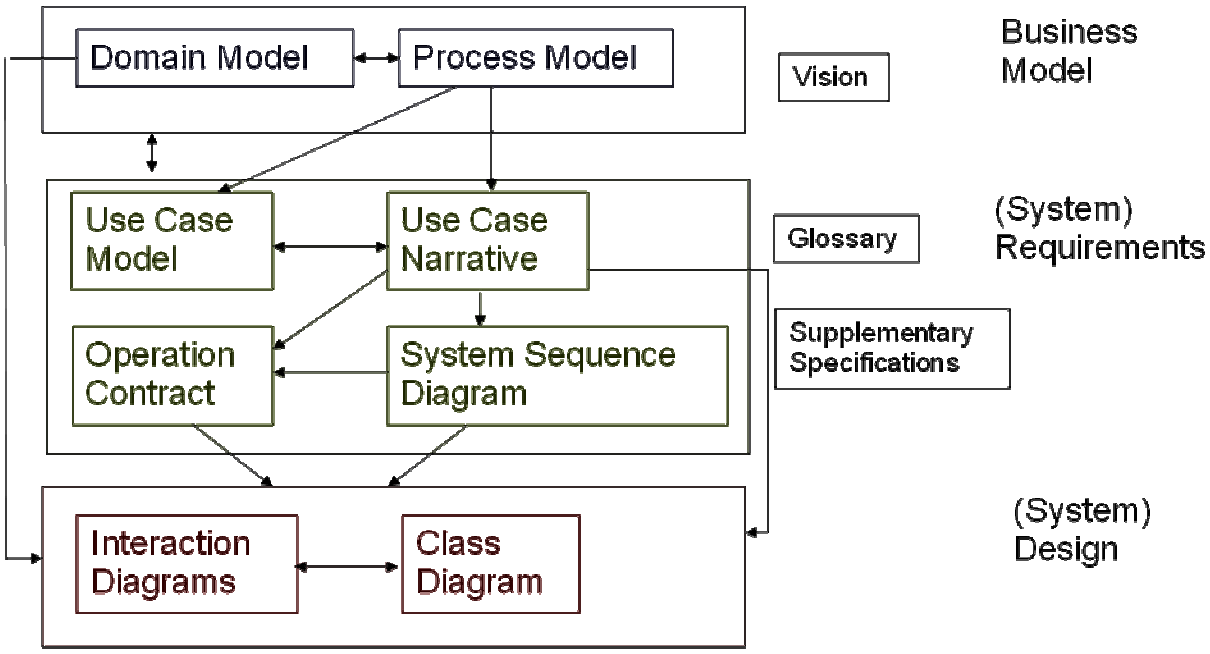

Figure 2. Conceptual connections between modeling artifacts in Larman's Agile Unified Process

The global aspects of systems development need to be addressed in some course, but based on the breadth of the issues already slated to the SA&D course, we recommend that it is not the likely candidate for covering outsourcing issues. Certainly, students need to learn the capabilities of companies in key vendor countries. They also need to examine an abbreviated contract between the client and the vendor company. Although most students are aware that coding is commonly outsourced, the distribution of other life cycle activities among the client and the vendors is rarely discussed. For example, can testing be outsourced? Can certain kind of design be outsourced? Can certain elements of analysis be outsourced? These issues are complex, and it does not seem that these can be covered in a regular SA&D course.

An interesting side effect is that global distributed development can run counter to agile development [Batra, 2007]. For example, agile principles recommend face-to-face meetings and tacit understanding among team members; these requirements are not practical in distributed development. Agile principles prescribe satisfying customer requirements that are even requested very late in the project, while distributed development is governed by well-documented contracts. In summary, the SA&D course is not geared to handle the complex issues that arise in a distributed development environment.

V. ELEPHANTS IN THE ROOM

The panel wanted to raise three questions regarding our collective capabilities and resources that potentially have the status of the proverbial elephant in the room. The first question is related to the number of IS faculty members who have strong enough software development capabilities to teach systems design. The second question relates to the difficulty in teaching agile methods without focusing on coding. The third question is the seemingly inherent conflict between the increasing publishing pressures that are present at all universities at this point and the faculty

members' need to stay technically up-to-date in order to be able to address all aspects of an iterative life cycle.

First, the "D" in SA&D refers to design, and although design can refer to activities at various levels, ultimately "design" in SA&D has to include the internal design of software solutions, such as the object/class structure and object responsibility design in object-oriented design. These design activities are neither technically trivial nor something that can be ignored. No reliable statistics known to us are available, but anecdotal evidence suggests that many IS faculty members teaching SA&D courses are not comfortable with actual software design issues – in many SA&D textbooks, "design" includes only user interface and database design, and it appears that in many cases, faculty members are comfortable only with these two aspects of design.

Second, iterative lifecycle models (including UP, but particularly the agile models) by definition include coding as one of the core activities during all iterations. The role of coding is valued even in projects based on the structured approach (Kohli and Gupta, 2002). Therefore, it is improbable to teach the iterative approach and particularly apply it to a project unless the instructor has sufficient capabilities not only in modeling but also in guiding students' coding work at a high professional level. Again, evidence is scarce, but anecdotally, it appears that many IS professors are intentionally staying away from activities close to the actual creation of a software artifact. If that is the case, it is not clear to us how one can effectively move between different SA&D activities as specified by the iterative models, particularly because learning from an actual functional software artifact is a critically important part of any iterative model (whether UP-based or a pure agile model).

Third, teaching any approach to SA&D that includes the creation of a functional artifact requires that the faculty member teaching the course is capable of supervising all parts of the process and guiding the students also in the process of actually creating the artifact. Do our reward mechanisms and promotion and tenure processes sufficiently acknowledge the fact that maintaining one's capabilities in software development is a highly time-consuming activity, which is not necessarily compatible with the type of research expected by the journals with the highest stature in our field? Is it, in practice, a feasible idea to expect a faculty member to publish at the highest level and teach SA&D courses using approaches that include software design in its true sense and with a strong focus on a functional software artifact?

It is very important that we acknowledge possible limitations of our field if they are a reality. If IS faculty members are not being trained in design and technical architecture of software solutions, and if it is not possible for an average IS faculty member to maintain his or her technical capabilities while producing scholarly output at an acceptable level, maybe we should limit our focus on SA&D activities that do not address aspects of the design of the software solution. If, however, we believe that the "D" in SA&D is important, we have to both ensure that all faculty members teaching SA&D courses have a sufficient proficiency in design and that the time spent to maintains one's proficiency in this area is recognized. The affected SA&D faculty need to inform their chairpersons about this challenge, but we doubt an exception will easily be made. However, some of the other IS areas (e.g., information security, data communications, data mining) are likely to face similar issues, and we hope the increased awareness of the issue may lead to an acceptable solution.

VI. CONCLUSIONS

This paper discussed the need to reexamine the SA&D course in the IS curriculum to better address the trends affecting analysis and design. One of the clearest messages from the audience at the end of the panel was the need to identify and focus on the core immutable principles and skills of SA&D. There was not enough time to discover whether or not there was consensus among the audience members regarding the nature of these principles or skills, but there was a clear call for us as a field to identify these and advocate them as the core intellectual content of the SA&D course(s).

We are prepared to make ten specific recommendations based on our analyses and on the input from the panel participants. Although there is still room for debate, each member of the IS community involved with teaching SA&D or coordinating the curriculum with the SA&D course should consider these recommendations:

1. Only include traditional structured analysis techniques (e.g., data flow diagrams) for historical context if desired, but not in any depth. Instead, establish use case modeling as the new functional requirements standard.
2. Focus on UML as the primary modeling grammar.
3. Cover only a subset of UML in the SA&D course. Activity diagrams, use case descriptions and diagrams, class diagrams, and sequence diagrams are typically sufficient.
4. Help students understand how a modeling grammar (such as UML class diagrams) can be used for multiple purposes. For example, UML class diagrams can be used for modeling both the problem domain and the internal structure of the software solution.
5. Use the core Unified Process life cycle model (including the separation of Disciplines and Phases) or a variant as the organizing framework for SA&D concepts. Use this framework to help students understand that the disciplines can be configured into a variety of different processes.
6. Cover agile values and principles in the course, but don't use radical versions as the philosophy underlying the course project.
7. Emphasize the internal consistency of the modeling artifacts, showing, for example, how a use case is designed using a sequence diagram and a sequence diagram establishes the methods added to the class diagrams.
8. If possible, integrate the SA&D course closely with a software project management course with a global focus. Assign the issues related to distributed development in the project management course.
9. Acknowledge the fact that the design of the internal structure of software is a complex activity that requires in-depth understanding of software development.
10. As an academic field, we have to help department chairs and deans understand that teaching SA&D with an approach that includes design and implementation is difficult and very resource intensive.

ACKNOWLEDGEMENTS

The authors wish to thank the participants at the AMCIS 2007 Panel for their interest and contributions to the discussion and to this paper.

REFERENCES

Bajaj, A., Batra, D., Hevner, A., Parsons, J., and Siau, K. (2005) "Systems Analysis and Design: Should we be researching what we are teaching," *Communications of the AIS*, Volume 15, pp. 478-493.

Bergstrom, S. and Raberg, L. (2004). *Adopting the Rational Unified Process*. Boston, MA: Pearson Education.

Batra, D. (2007) "Modified Agile Practices for Outsourced Software Projects," forthcoming in the *Communications of the ACM*.

Batra, D. (2008) "Unified Modeling Language (UML) Topics: The Past, the Problems, and the Prospects," forthcoming in the *Journal of Database Management*.

Beck, K. (2000). *Extreme programming explained: Embrace change*. Reading, MA: Addison-Wesley.

Boehm, B. W., & Turner, R. (2003). *Balancing agility and discipline: A guide for the perplexed*. Boston: Addison-Wesley.

Cockburn, A. (2001). *Writing effective use cases*. Reading, MA: Addison-Wesley.

Cockburn, A. (2005). *Crystal clear: A human-powered methodology for small teams*. Boston: Addison-Wesley.

Cullen, S. and Wilcocks, L. (2003). *Intelligent IT Sourcing: Eight Building Blocks to Success*. Burlington, MA: Elsevier Butterworth-Heinemann.

Dobing, B. & Parsons, J. (2008). "Dimensions of UML diagram use: A survey of practitioners," *Journal of Database Management*, *19*(1) (forthcoming).

Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (3rd ed.). Boston, MA: Addison-Wesley.

Gorgone, J. T., Gordon B. Davis, Joseph S. Valacich, Heikki Topi, David L. Feinstein, and Herbert E. Longnecker. (2002). "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, pp. 1-52.

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

Iivari, J., Parsons, J., and Hevner, A.R. (2005) "Research in Information Systems Analysis and Design: Introduction to the Special Theme Papers," *Communications of the AIS*, Volume 16, 810-813.

Johnson, R. (2002) "Object-Oriented Systems Development: A Review of Empirical Research," *Communications of the AIS*, Volume 8, 65-81.

Kobryn, C. (1999) "UML 2001: A standardization odyssey," *Communications of the ACM, 42*(10), 29-37.

Kobryn, C. (2004). "UML 3.0 and the future of modeling," *Software & Systems Modeling, 3*(1), 4-8.

Kohli, R. and J.R.D. Gupta (2002). "Effectiveness of the Systems Analysis and Design Education: An Exploratory Study," *Journal of End User Computing*, 14(3), 16-31.

Lang, M. (2006). "New branches, old roots: A study of methods and techniques in Web / hypermedia systems design," *Information Systems Management,* 23(3), 62-74.

Larman, C. (2004) *Agile & Iterative Development: A Manager's Guide*, Boston, MA: Addison-Wesley.

Larman, C. (2005). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development* (3rd ed.). Upper Saddle River, NJ: Prentice Hall PTR.

Mahapatra, R., Nerur, S., and Slinkman, C. (2005) "Teaching Systems Analysis and Design – A Case for the Object Oriented Approach," *Communications of the AIS*, Volume 16, 848-859

Olson, J. S., & Olson, G. M. (2003). "Culture surprises in remote software development teams," *ACM Queue*, 1(9), 52-59.

Ragowsky, A., Somers, T.M., and Adams, D.A. (2005) "Assessing the Value Provided by ERP Applications through Organizational Activties," *Communications of the AIS*, Volume 16, 381-406.

Reeves, W. W. (1999). *Learner-Centered Design: A Cognitive View of Managing Complexity in Product, Information, and Environmental Design*. Thousand Oaks, CA: Sage Publications.

Satzinger, J.W. and Jackson, R.B. (2003) "Making the Transition from OO Analysis to OO Design with the Unified Process," *Communications of the AIS*, Volume 12, 659-683.

Schwaber, K., & Beedle, M. (2002). *Agile software development with scrum*. Upper Saddle River, NJ: Prentice Hall.

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

Siau, K. & Loo, P.-P. (2006). "Identifying difficulties in learning UML," *Information Systems Management, 23*(3), 43-51.

Topi, H., Valacich, J., Nunamaker, J., Sipior, J. (2007) "Undergraduate Information Systems Model Curriculum Revision: Rethinking the Approach and the Process," *Proceedings of the AMCIS 2007*, Keystone, Colorado.

ABOUT THE AUTHORS

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*.

Dinesh Batra is a Knight-Ridder Research 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, European Journal of Information Systems, Communications of the AIS, International Journal of Human Computer Studies, Computers and Operations Research, Data Base, Information and Management, Journal of Database Management, Decision Support Systems, Requirements Engineering Journal,* and others. He has served as the President of the AIS SIG on Systems Analysis and Design.

Heikki Topi is Associate Dean of Graduate and Executive Programs at Bentley College. His research has been published in journals such as *European Journal of Information* Systems, *JASIST*, *Information Processing & Management*, *International Journal of Human-Computer Studies*, *Journal of Database Management*, *Small Group Research*, and others. He has been actively involved in national computing curriculum development and evaluation efforts (including *IS2002* and *CC2005 Overview Report*), and he is a member of the ACM Education Board and co-chair of the current IS curriculum revision project.

Communications of the Association for Information Systems

ISSN: 1529-3181