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

Systems Analysis & Design: An Essential Part of IS Education

Albert L. Harris
Department of Computer Information Systems
Appalachian State University
Boone, NC 28608, USA
harrisal@appstate.edu

Michael Lang
Business Information Systems Group
National University of Ireland, Galway
University Road, Galway, Ireland
Michael.Lang@nuigalway.ie

Briony Oates
School of Computing
University of Teesside
Middlesbrough, TS1 3BA, UK
B.J.Oates@tees.ac.uk

Keng Siau
Department of Management
209 College of Business Administration
University of Nebraska-Lincoln
Lincoln, NE 68588-0491, USA
ksiau@unlnotes.unl.edu

ABSTRACT

Systems analysis and design has been as critical building block in Information Systems (IS) education since the inception of the IS major. Whether it is taught using the traditional or "structured approach or the object-oriented approach, it exposes students to the different methods, tools, and techniques used in developing new systems, develops students analytical and problem-solving skills, teaches fact-finding and data gathering techniques, and provides teamwork skills. All of these are valuable skills for systems analysts. This paper introduces the reader of this special issue on systems analysis and design education to the issue, discusses the two approaches to teaching systems analysis and design, and ways that it is taught in the various curricula.

Keywords: Systems Analysis & Design, IS Curriculum, Teaching Approaches

1. INTRODUCTION

A course in systems analysis and design has been a critical part of the Information Systems (IS) curriculum ever since the inception of the IS major. Most educators see systems analysis and design (SA&D) as a required building block for IS students (Bajaj et al., 2005). It appears as a core course for IS majors and minors in the IS 2002 model undergraduate curriculum (Gorgone et al., 2002) and as part of the core of the MSIS 2000 model graduate curriculum (Gorgone et al. 2000). However, if you were to assemble 50 IS professors into one room, you would likely end up with 50 different

opinions about how best to teach SA&D. The one thing that they would probably all agree on is that it is a very challenging course to teach.

The most effective method to teach a course in systems analysis and design is dependent upon many factors, including, but not limited to: the instructor's experience in SA&D, the positioning of the course within the overall curriculum, the profile of the class, the modeling approach taken (e.g. structured or object-oriented), and the needs of employers, just to name a few. In some curricula, it sets the stage for more advanced courses in database and application development. In other programs, it is a capstone course that integrates prior coursework using an advanced project. No matter what teaching method is used, there is general agreement that systems analysis and design is an essential part of IS education.

2. WHY IS IT ESSENTIAL?

Throughout the evolution of the IS Model Curriculum, systems analysis and design skills have been a cornerstone of the curriculum. According to the 2002 Model Curriculum, "Systems analysis provides experience determining system requirements and developing a logical design. Instruction in physical design of information systems will ensure that the students can use a logical design to implement information systems in both a DBMS and in emerging development environments" (Gorgone et al., 2002).

For a number of important reasons, a course in SA&D is an indispensable component of any program of study in information systems. These reasons include:

- It exposes students to the different methods, tools, and techniques used in developing new systems – Information systems analysis and design in the "real-world" is an inherently complex task, fraught with many difficulties and challenges. There is no universally applicable single-best-way of approaching this task. Some organizations follow a standardized published method, others develop their own in-house method, and others have no pre-specified method. In practice, analysts often tailor methods or combine fragments of different methods and techniques, as appropriate to the problem at hand. This has been termed "method-in-action" (Fitzgerald, Russo, and Stolterman, 2002) or "improvisation" (Ciborra, 1999). Accordingly, it is necessary to equip students with a toolbox of methods and techniques drawn from different sources, and moreover to impress upon students how the same problem can be approached in different ways using different combinations of these methods and techniques, which in turn may be influenced by one's tacit philosophical beliefs and ideological persuasions.
- It develops analytical and problem-solving skills – The very first task of a systems analyst is to define the problem to be solved by the new system. Therefore, analysts must possess analytical and problem solving skills. A course in SA&D is a means of developing and sharpening students' analytical and problem solving techniques, as well as encouraging them to think critically and to be creative.
- It teaches fact-finding and data gathering techniques – Fact finding and data gathering are fundamental skills of systems analysts and are used heavily in the systems analysis activities. Techniques such as conducting interviews, sampling, developing questionnaires, and documenting data flows are normally taught in the SA&D class.
- It provides teamwork skills – The systems analysis and design process in an organization is a team effort. As a result it is often taught by means of getting small groups of students to work together. This provides an excellent platform for students to learn teamwork skills such as project management, task co-ordination, group communication, brainstorming, pair analysis & design (e.g. Extreme Programming), and synchronous or asynchronous collaboration. Students, just like IS professionals in an organization, must learn to work together to develop a good system for the users.
- It is a necessary prerequisite for any course in database development or applications programming – However adept a programmer may be in code production, he must also be able to understand and produce requirements specifications. Programmers and software engineers who think only in terms of technical solutions might produce systems that are technically sound, but which are woefully inappropriate or unworkable in practice (e.g. because important operational constraints and "soft" socio-technical issues are overlooked, or *realistic* user interaction scenarios are not given adequate forethought). Programmers who want to jump straight to coding may end up solving the wrong problem, or perhaps developing a solution that actually gets in the way (e.g. excessively rigorous or inflexible rules hard-coded into a system can lead to frustrated end-users; how often do we hear customer service operators or MIS staff apologizing that "I'm sorry, but the system won't let me ..."). A course in SA&D encourages trainee systems developers to think in terms of user needs, organizational context, business value-added, and prioritized functional requirements.
- Above all, systems analysis and design is critical to the successful development and implementation of information systems. Poorly designed systems are ineffective, inefficient, wasteful of resources, and likely to cause user dissatisfaction. In particular, requirements analysis is of paramount concern. Brooks (1987) perhaps expressed it best: "the hardest single part of building a software system is deciding precisely what to build ... No other part of the work so cripples the resulting system if done wrong". Vague and ambiguous requirements at best make it difficult to prepare accurate time and cost estimates. At worst, they lead to misdirected development effort and ultimate end-user rejection. For wide-audience or public information systems such as e-government, e-banking, mobile commerce, and most Web-based systems, it is imperative that a system should be designed so that it is robust, reliable, secure, and easy to use. This requires a

sound knowledge of the principles of real-time interactive systems design, and additionally of security modeling, user-centred design, and applications programming (it is hard to specify a design requirement without having at least some idea of how it might be technically implemented).

The job of the systems analyst is to define the business needs of the client (which may involve such activities as strategic systems planning, audience definition, or business area analysis) and then to design a feasible information system that best meets those needs (which requires technical knowledge, domain knowledge, and a sensitive awareness of operational issues). A systems analyst can find himself/ herself having to fill various roles: business analyst, systems architect, organizational change agent, political negotiator, project manager, and joint application design (JAD) facilitator to name a few. It is a duty that carries a heavy burden of responsibility and expectation, and for which a range of skills, – technical, business, and interpersonal, – is required. The curricula for IS programs therefore needs to be purposefully designed to provide a balanced coverage of these various aspects of systems analysis and design.

3. CONTENT OF SYSTEMS ANALYSIS AND DESIGN COURSES

Simply put, systems analysis and design is an approach to the development of information systems which encompasses the first four phases of the systems development life cycle (SDLC) – Planning, Analysis, Design, and Implementation. The SA&D process can encompass many tools and techniques. Broadly speaking, there are two main modeling approaches to SA&D:

- The traditional or "structured" approach, which uses Data Flow Diagrams (DFDs) and Entity-Relationship Diagrams (ERDs) as its modeling tools;
- The object-oriented (OO) approach, for which the Uniform Modeling Language (UML) has become the *de facto* standard.

The traditional approach to SA&D is generally considered process-centric and top-down, in so far as the problem under consideration is decomposed into a hierarchical set of processes. In the traditional approach, the systems analysis phase consists of all activities needed to understand the system and specify in detail what the system is to do. The systems design phase consists of all activities needed to specify the solution and how it will be physically implemented.

The object-oriented approach is generally considered data-centric. It uses a set of entities (or more correctly, "classes") that encapsulate both the data ("attributes") and processes ("methods") associated with each entity type.

3.1 Traditional Approach

The traditional approach to SA&D adopts a formalized step-by-step approach to the systems development life cycle (SDLC). It uses phases and activities; the activities of one phase must be completed before moving to the next phase. At the completion of each activity or phase, a document is produced that must be approved by the stakeholders before moving to the next activity or phase. The structured approach looks at a system from a top-down view, and it uses separate data and process models. The technique of data flow diagramming (DFDs, sometimes DFMs – data flow models) is used to depict the business processes of a system and the information that flows between and within processes. The DFDs and their associated data dictionary contain information about the inputs, outputs, processes, and data storage that need to be designed and ultimately built. For data modeling, entity-relationship diagrams (ERDs) are used.

Teaching the traditional approach has the advantage that it has been around for many years and most SA&D texts describe the SDLC "waterfall" model in detail. The chapters of these texts logically follow the phases and activities of the SDLC (e.g. Dennis and Wixom, 2000; Hoffer, George, and Valacich, 2002; Kendall and Kendall, 2001; Marakas, 2001; Satzinger, Jackson, and Burd, 2004; Shelly, Cashman, and Rosenblatt, 2003; Whitten, Bentley, and Dittman, 2004). Training in traditional approach is also needed for analysts and programmers to maintain information systems that were developed in the past 10-20 years using the traditional approach. The negative aspect is that many new systems will be developed based on the object-oriented approach, which uses quite a different way of formulating and resolving problems. Also, many CASE tools are moving away from or are no longer supporting DFD and ERD, and this causes problems in teaching. Convincing students that the traditional approach is still relevant and worth learning is also a challenge.

3.2 Object-Oriented Approach

The object-oriented (OO) approach was initially developed by software engineering professionals dealing with large and complex systems in domains such as aerospace and process control (Schach, 2004). The object-oriented approach takes a bottom-up approach to systems development. It describes the system through a set of business processes and the object classes that these processes deal with. It uses a set of interconnected diagrams to represent the views and functionality of a system. Though there are many alternative notations for OO diagrams, the most common standard now used in industry is UML. Beyond these OO modeling techniques, there exist systems development methods and approaches such as the Unified Process (Satzinger, Jackson, and Burd, 2004). In general, object-oriented analysis and design (OOAD) follows an iterative and incremental approach to systems development. The systems development process is viewed as a series of increments or phases: inception, elaboration, construction, and transition (Booch, Jacobson, and Rumbaugh, 1999). In each increment or phase, the developers move through the activities of gathering requirements, analyzing the requirements, designing the system, implementing the design, and testing the system.

According to the UML creators (Booch, Jacobson, and Rumbaugh, 1999), object-oriented SA&D must have three characteristics:

1. Use case driven – A use case depicts the interactions between an information system and its users (e.g. human end-users, or other systems or agents). A use case diagram (an object-oriented technique) graphically depicts who will use the system and how the user expects to interact with the system.

2. Architecture-centric – This is the underlying software architecture of the new system. Object-oriented SA&D usually will encompass three separate, but interrelated, views of a system: functional, static, and dynamic.

3. Iterative and incremental – Because a typical system goes through continuous re-evaluations and modifications, the UML diagrams will be reviewed on a recurring basis and changes will generally result in incremental improvements.

With the UML, the object-oriented community has at least agreed on a standard modeling language. This resolves the issue of different instructors teaching different object-oriented modeling languages. Nevertheless, UML is huge and complex. UML 1.x has nine different diagrams. The study by Siau and Cao (2001) shows that UML 1.x is two to eleven times more complex than other object-oriented methods. UML 2.0 has thirteen different diagrams! (Dennis, Wixom, and Teagarden, 2005). The large number of diagrams is a concern for instructors as it is almost impossible to cover all of them in one semester. The paucity of good and systematic textbooks is a concern. Because of the newness of UML, some textbooks are also riddled with errors and inconsistencies.

3.3 Other Potential Materials for Systems Analysis and Design Courses

Many advanced SA&D courses go beyond just teaching modeling approaches. They also explain the process by which different models are combined or validated against each other (Satzinger, 2006). They may discuss different process models, such as the conventional "waterfall" model of the SDLC, incremental and evolutionary approaches, rapid application development (RAD), and the spiral model (Boehm, 1988). They might also introduce students to the Capability Maturity Model (Paulk et al., 1993) and to some of the standardized published methods, such as Information Engineering (Martin, 1989), the Rational Unified Process (Kruchten, 2000) and Soft Systems Methodology (Checkland and Scholes, 1990). The most advanced courses focus on the theoretical underpinnings of the various modeling approaches and methods, looking at their inherent ontological and epistemological assumptions (e.g. Hirschheim, Klein, and Lyytinen, 1995).

In addition, new contemporary developments in the world of practice provide new materials for SA&D courses, particularly those at advanced levels. Agile methods, open source software (OSS), globally-distributed software development, commercial-off-the-shelf (COTS) applications configuration, and component-based systems development are relatively new kids on the block that are gaining in popularity. New applications related to Web-based and multimedia systems, geographical information systems (GIS), enterprise resource planning (ERP) systems, ubiquitous computing, bio-informatics, e-government, mobile commerce and other advanced technologies may require different approaches to systems analysis and design. There is also an "amethodical" movement that advocates a very different approach to systems development (e.g. Introna and Whitley, 1997; Truex, Baskerville, and Travis, 2000). The more advanced topics discussed above are usually studied at graduate level.

4. WAYS OF TEACHING SYSTEMS ANALYSIS AND DESIGN

So, how should we approach the teaching of SA&D? Take any ten IS curricula, and you might see ten different ways that it is taught. In many universities, coverage of SA&D consists of a single course, generally taken in the Junior year. Although comparatively rare, some programs split the subject into two courses: Systems Analysis, and Systems Design. Additionally, most programs have an advanced second course in SA&D, often referred to as the "capstone" course. This is usually taken in the last semester of a program of study and focuses on an actual systems development project, often dealing with an applied "real world" case study involving the design and development of a system for actual users. Some schools are also experimenting with "apprenticeship" or "design studio" approach. This approach is very "hands-on" in orientation and students typically work on real-world projects for one to two years while taking courses at the same time. Instructors and mentors work closely with the students on the projects, which are funded by organizations.

Another important issue is: should a SA&D class be taught using the traditional approach, or the object-oriented approach, or both? With current trends, many IS educators feel that students need to understand the key object-oriented concepts and technology so as to be better prepared for the future (Rob, 2006). But whereas some educators have completely embraced the object-oriented approach in their courses, others seem determined for whatever reason to completely ignore it. Some schools teach a course on SA&D based on the traditional approach, followed by another based on the object-oriented approach. Some teach them in the opposite order. If only one course is taught, which approach should we be teaching? Can we combine both approaches into one course? If so, how can we best fit the object-oriented topics with the existing coherent discussion of the structured approach? Can OO methodology be included along with the traditional approach in the same text? Are the newer techniques and methodologies difficult for traditional analysis and design instructors to adopt? Do students have difficulty learning and using the newer techniques? Is there a lag in the diffusion of newer analysis and design techniques and methodologies in the classroom? If so, are we as educators failing to meet the needs of industry with regard to desired graduate skills?

New ways of teaching SA&D are also emerging, such as internships, apprenticeships, project-based learning, and

service-based learning. Are these more effective than our current pedagogical approaches? As students move to more advanced study of systems analysis and design, how can we give them a realistic experience of published methods within the limited time available? How can we develop students' critical skills to analyze, compare and evaluate different methods? How do we enable technically-oriented students to tackle philosophical concepts about the nature of 'reality'?

As a field matures, there are more and more topics to cover. Systems analysis and design is no exception! How can we teach all these topics without increasing the credit hours that a student needs to have in learning systems analysis and design? The current trend in higher education is to reduce the number of credit hours it takes for a student to graduate (both undergraduate and graduate programs). This is in conflict with the increasing number of topics and areas that a student is expected to learn before graduating. With most of the IS programs (at least in the US) residing in the business schools, there are only so many credit hours that can be allocated to IS courses after a student fulfills his/her general education (i.e., core courses for a university student) and business requirements (i.e., core courses for a business student).

These are some of the issues and questions facing SA&D educators and the IS discipline. Possible answers to some of them are presented in the papers in this special issue.

5. PAPERS IN THIS ISSUE

We assert that SA&D should be an essential component of any IS program because it is concerned with critical aspects of information systems development and implementation, and is therefore at the very core of the discipline of information systems. However, there has been something of a teaching-research gap within the area of SA&D. A survey of the ISWorld Faculty Directory found that 22% indicated systems analysis and design as one of their teaching areas, yet SA&D is not well represented if you look at the number of articles appearing in the main IS/IT journals and conferences in recent years (Bajaj et al., 2005). We are therefore greatly pleased to introduce this special issue of *Journal of Information Systems Education (JISE)* on the theme of "Systems Analysis and Design Education". The call for papers for this issue invited submissions through two channels, either directly to *JISE* or indirectly by being "fast-tracked" as a selected paper from the *1st AIS SIGSAND European Symposium on Systems Analysis & Design* (held in Galway, Ireland in June 2006, for which the guest editors served as program chairs). We were pleased to receive a total of 40 submissions, which reflects a healthy interest in the topic of SA&D education. The articles selected for inclusion in this special issue include an invited paper and 9 additional papers chosen on the basis of overall merit. This sample of papers, though small, is quite varied in terms of the issues addressed and is indicative of the range of topics which are broadly of interest to teachers and researchers within the field of SA&D.

This Special Issue on systems analysis and design starts with a Teaching Tip titled "Interleaving Modeling and Writing Activities in Systems Analysis and Design by James J. Pomykalski. This is followed by an invited paper titled "Reflections on Teaching Information Systems Analysis and Design: From Then to Now." In this invited paper, Avison, Cole and Fitzgerald contemplate their collective experiences over a time span of three decades, extending from the early days of the structured methodologies era in the 1970s to the present time. They provide a number of helpful suggestions on the efficacy of course delivery methods based on their own experimentation, and stress that IS education should seek to instill lasting skills and knowledge that withstands passing fashions, paradoxically noting that "the material changes all the time while staying the same".

The two papers by Batra and Satzinger ("Contemporary Approaches and Techniques for the Systems Analyst") and Bataveljic, Eastwood and Seefried ("An Approach to Teaching Object-Oriented Systems Analysis and Design") both address the important issue of how to usefully combine structured and object-oriented models, methods and techniques within the same curriculum (e.g. RUP, SDLC, UML, DFDs). While there is much common ground, some notable differences of opinion also emerge. Most notably, whereas Bataveljic and colleagues argue that data flow modeling should be retained because it facilitates process decomposition and assists use case definition, Batra and Satzinger argue that it is no longer appropriate for the modern object-oriented development environment and should therefore be abandoned. Both papers make a number of provocative recommendations and observations which are likely to be of much interest and relevance to SA&D educators grappling with the issue of how to best mix the old with the new.

Preiser-Houy and Vanarrete present a very interesting paper on the application of a service-learning approach to SA&D education ("Exploring the Learning in Service-Learning: A Case of a Community-based Research Project in Web-based Systems Development"). By immersing students within a community-based project, they are actively exposed to the realities of SA&D in practice and are thereby challenged to develop a variety of skills and knowledge, both general and domain-specific. The article classifies these skills within a theoretical framework under the banners of academic, personal, and inter-personal student learning outcomes. This paper contains a number of valuable lessons on the use of service learning and community-based research, and the framework presented can usefully serve as a template to assist the purposeful design of a course in SA&D or indeed any other applied social discipline.

The paper by Van Vliet and Pietron ("Information Systems Development Education in the Real World – A Project Methodology and Assessment") addresses two pertinent issues: (1) is the SDLC still a valid and relevant model for the purposes of SA&D education in the modern world?, and (2) how useful are real-world group projects in preparing graduates for the workplace? The findings of their survey of IS graduates reveal that traditional methods and techniques remain very popular in practice, and that the use of realistic student projects, though more taxing on staff time and resources, deliver substantial long-term educational benefits.

Steven Alter ("Pitfalls in Analyzing Systems in Organizations") presents an intriguing article on typical errors and omissions in systems analysis and design, based on an examination of over 200 term papers received by him from MBA students over a number of years. He classifies these pitfalls under nine different categories, – which include inadequate critical thinking, failure to define the system, viewing technology as the system, and failure to reflect upon organizational issues, – and prescribes a number of recommendations as to how these problems can be addressed in SA&D courses and textbooks.

Steve McRobb ("Challenging Students: Reflections on the Development and Delivery of an Undergraduate Module that Introduces the Full Systems Development Life Cycle") makes an interesting contribution to this special issue in the form of an experience report that discusses the introduction of an innovative module into the first year of an undergraduate IS program. This module challenges novice students to concurrently learn and integrate skills in systems analysis and design, applications programming, and Web development, the objective being to promote a holistic understanding. As one would expect, such an ambitious undertaking was not without its detractors and critics, and of experiences to date there are both negative and positive aspects to report. The paper sets forth some valuable lessons and recommendations which are likely to be of benefit to IS educators who might be considering a similar departure or are already along the way.

There are quite a few topics within the field of SA&D which cause no small degree of consternation amongst students when they first grapple with them. One such topic, which has long presented difficulties for both students and instructors, is database normalization. The highly original contribution by Kung and Tung ("An Alternative Approach to Teaching Database Normalization: A Simple Algorithm and an Interactive e-Learning Tool") is a welcome addition to the SA&D literature because it describes an alternative easy-to-use algorithm for generating consecutive normal forms. It also illustrates the application of an interactive Web-based e-learning tool which helps students to form an understanding of how to proceed from one stage to the next.

Akhilesh Bajaj ("Large Scale Requirements Modeling: An Industry Analysis, a Model, and a Teaching Case") discusses the topical SA&D issue of buy (e.g. COTS) versus build (i.e. custom-designed bespoke applications), and argues that the pendulum of general favor is swaying back towards a propensity to build. He argues that systems analysis and design, a field that lost some of its traditional ground with the emergence of readily available COTS solutions for business applications, is regaining importance and is set to play an increased role in requirements modeling for large-scale information systems. With this in mind, he describes an extended modeling technique called Entity Relationship Activity (ERA) and presents a teaching case to which it has been applied.

In producing a requirements specification document, systems analysts aspire to capture a set of requirements that are complete, consistent, and correct. With the emergence of lean and agile development approaches such as Extreme Programming (XP), Scrum, and Dynamic Systems Development Method (DSDM), there is a stronger emphasis on test-driven design and "getting it right" from the outset. Against this background, the teaching tip article by Craig Tyran ("A Software Inspection Exercise for the Systems Analysis and Design Course") is a timely contribution because it illustrates how principles of software inspection, traditionally not a part of systems analysis, can be applied to validate and verify design models.

6. REFERENCES

Bajaj, A., Batra, D., Hevner, A., Parsons, J. and Siau, K. (2005), "Systems Analysis and Design: Should We Be Researching What We Teach?", Communications of the AIS, Vol. 15, pp. 478-493.

Boehm, B.W. (1988), "A Spiral Model for Software Development and Enhancement", IEEE Computer, Vol. 21, No. 5, pp. 61-72.

Booch, G., Jacobson, I., and Rumbaugh, J. (1999), The Unified Modeling Language User Guide. Addison-Wesley, Reading, MA.

Brooks, F.P. (1987), "No Silver Bullet / Essence and Accidents of Software Engineering", IEEE Computer, Vol. 20, No. 4, pp. 10-18.

Checkland, P. and Scholes, J. (1990), Soft Systems Methodology in Action. John Wiley & Sons, Chichester.

Ciborra, C. U. (1999), "A Theory of Information Systems Based on Improvisation" In Currie, W. L. & Galliers, B. (eds), Rethinking Management Information Systems, pp. 136-155, Oxford University Press.

Dennis, A. and Wixom, B.H. (2000), Systems Analysis and Design. John Wiley & Sons, New York.

Dennis, A., Wixom, B.H. and Teagarden, D. (2005), Systems Analysis and Design with UML Version 2.0. John Wiley & Sons, New York.

Fitzgerald, B., Russo, N.L. and Stolterman, E. (2002), Information Systems Development: Methods in Action. McGraw-Hill, London.

George, J.F., Batra, D., Valacich, J.S. and Hoffer, J.A. (2004), Object-Oriented Systems Analysis & Design. Prentice Hall, Upper Saddle River, NJ.

Gorgone, J.T., Gray, P., Feinstein, D.L., Kasper, G.M., Luftman, J.N., Stohr, E.A., Valacich, J.S., and Wigand, R.T., (2000), "MSIS 2000 Model Curriculum And Guidelines For Graduate Degree Programs In Information Systems", Communications of the AIS, Vol. 3, No 1.

Gorgone, J.T., Davis, G.B., Valacich, J.S., Topi, H., Feinstein, D.L., and Longenecker, H.E. (2002), "IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems", Communications of the AIS, Vol. 11, No 1.

Hirschheim R.A., Klein, H.-K., and Lyytinen, K (1995), Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations. Cambridge University Press.

Hoffer, J.A., George, J.F., and Valacich, J.S. (2002), Modern Systems Analysis & Design, 3rd ed. Prentice Hall, Upper Saddle River, NJ.

Introna, L.D. and Whitley, E.A., (1997), "Against Methodism: Exploring the Limits Of Method", Information Technology and People, Vol. 10, No. 1, pp. 31-45.

Kendall, K. and Kendall, J. (2002), Systems Analysis and Design, 5th ed. Prentice Hall, Upper Saddle River, NJ.

Kruchten, P. (2000), Rational Unified Process. An Introduction. Addison-Wesley, Reading, MA.

Marakas, George M. (2001), Systems Analysis and Design: An Active Approach. Prentice Hall, Upper Saddle River, NJ.

Martin, J. (1989), Information Engineering. Prentice Hall, Englewood Cliffs, NJ.

Paulk, M. Curtis, B., Chrissis, M. and Weber, C. (1993), "Capability Maturity Model for Software, version 1.1", IEEE Software, Vol. 10, No. 1, pp. 18-27.

Rob, Mohammad A. (2006), "Dilemma Between The Structured and Object-Oriented Approaches to Systems Analysis and Design", Journal of Computer Information Systems, Vol. 46, Issue 3, Spring 2006, pp. 32-41.

Satzinger, John W. (2006), "Moving the Analysis and Design Course into the Future", Thomson Course Technology: Tech Trends - Industry Articles. http://www.course.com/techtrends/systems_analysis_042000.cfm, Accessed 9/8/2006

Satzinger, J.W., Jackson, R.B., and Burd, S.D. (2004), Systems Analysis and Design in a Changing World, 3rd ed. Course Technology, Boston, MA.

Schach, S.R. (2004), Introduction to Object-Oriented Analysis and Design with UML and the Unified Process. Irwin-McGraw Hill, New York.

Shelly, G.B, Cashman, T.J., and Rosenblatt, H.J. (2003), Systems Analysis and Design, 5th ed. Course Technology, Boston, MA.

Siau, K. and Cao, Q. (2001), "Unified Modeling Language (UML) - A Complexity Analysis", Journal of Database Management, Vol. 12, No. 1, pp. 26-34.

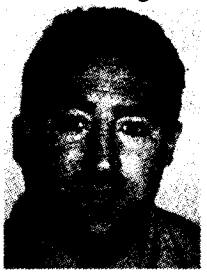
Truex, D., Baskerville, R. and Travis, J. (2000), "Amethodical Systems Development: The Deferred Meaning of Systems Development Methods", Accounting, Management & Information Technologies, Vol. 10, No. 1, pp. 53-79.

Whitten, J.L., Bentley, L.D., and Dittman, K.C. (2004), Systems Analysis and Design Methods, 6th ed. McGraw Hill, New York.

AUTHOR BIOGRAPHIES

Albert L. Harris is a Professor in the Department of Information Technology and Operations Management at the John A. Walker College of Business, Appalachian State University and Editor of the Journal of Information Systems Education, the leading international journal on information systems education. He is a Certified Management Consultant (CMC), a Certified Information Systems Auditor (CISA), and a Certified Systems Professional (CSP). He received his Ph.D. in MIS from Georgia State University, his M.S. in Systems Management from the George Washington University, and his B.S. in Quantitative Business Analysis from Indiana University. Dr. Harris teaches a variety of graduate and undergraduate classes in information systems. He served for three years as Acting Chair of the Department of Information Technology and Operations Management. He is a member of the Board of Directors for the International Association of Information Management and the Education Special Interest Group of AITP. He has served as Treasurer of Education Special Interest Group of AITP and Secretary for the Southeast Chapter of the Decision Sciences Institute. He has consulted for numerous private and public organizations, both in the United States and in several countries in Europe and Central America. He has published in the *Journal of Management Consulting, Information & Management, Journal of Information Systems Education, Journal of Computer Information Systems, Journal of Computer and Mathematics Education, Computerworld,* and numerous international, national, and regional conference proceedings. Dr. Harris has traveled extensively and has used these experiences in his teaching and research. Prior to becoming an educator and researcher, he spent almost 15 years in IT consulting, the last five managing his own consulting firm.

Michael Lang has been a Lecturer in Information Systems at National University of Ireland, Galway since 1996. His main research interest is business systems analysis & design, especially Web-based systems development, in which area his work has appeared in *Communications of the AIS, IEEE Software, IEEE Multimedia, Information Systems Management, Requirements Engineering,* and *Information & Software Technology*. He received his PhD from the University of Limerick and his MSc from NUI Galway, and presently serves as the European Co-coordinator for the AIS Special Interest Group on Systems Analysis & Design (SIGSAND).



Briony J. Oates is Reader in Information Systems at the University of Teesside, Middlesbrough, UK, where she gained her PhD in Information Systems in 2000. She has published around 80 papers in IS conferences and journals on social informatics, systems development methods, and research methods for IS. As a teacher she specializes in teaching advanced courses in systems analysis and design. She is also the author of a well-reviewed book aimed at masters and PhD students, *Researching Information Systems and Computing*, which was published in 2006 by Sage.

Keng Siau is a Professor of Management Information Systems (MIS) at the University of Nebraska-Lincoln (UNL). He is the Editor-in-Chief of the Journal of Database Management and the Series Editor of "Advances in Database Research." He received his PhD degree from the University of British Columbia (UBC) where he majored in Management Information Systems and minored in Cognitive Psychology. He has published more than 80 journal articles and these articles have appeared in journals such as *MIS Quarterly*, *Communications of the ACM*, *IEEE Computer*, *Information Systems*, *IEEE Transactions on Information Systems, Man and Cybernetics*, *IEEE Transactions on Information Technology in Biomedicine*, *IEEE Transactions on Professional Communication*, *IEEE Transactions on Education*, *DATA BASE*, *International Journal of Human-Computer Studies*, *IEICE Transactions on Information and Systems*, *Quarterly Journal of Electronic Commerce*, and others. In addition, he has published over 90 refereed conference papers in proceedings such as ICIS, HICSS, ECIS, WITS, and AMCIS. He has edited more than 10 books and written more than 15 book chapters. He served as the Organizing and Program Co-Chairs of the International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD) from 1996 to 2005. He was on the Organizing Committee of AMCIS 2005 and is serving on the Organizing Committee of AMCIS 2007.

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