Teaching the Integration of Information Systems Technologies

Jeffrey V. Nickerson, *Member, IEEE*

***Abstract*—A curriculum for a master's level course called Integrating Information Systems Technologies is presented. The course content is drawn from industry practice, and teaches ways of integrating large scale information systems. The teaching method used is similar to the method used to teach architecture. Students are asked to design complex systems in the form of posters; in each class these posters are discussed. Course evaluations from 95 students suggest that students find posters more effective than written papers and slide presentations. Specifically, students find that posters not only encourage visualization, but also provide them a way of seeing the thought processes of fellow students.**

***Index Terms*—Decision making, design evaluation, design science, information systems education, technical integration.**

I. Introduction

THE design of information systems is traditionally taught in a layered fashion. Students learn about networks, databases, and applications in separate courses. In management schools, students also take courses on finance, project management, and organizational behavior. This layered approach allows for the partitioning of material into areas of instructor expertise. However, practicing professionals in companies and government institutions often face situations in which all of these different layers need to be combined.

An information systems infrastructure for a large corporation includes networks, databases, messaging systems, and applications. Designing or modifying an infrastructure calls for an understanding not only of technology, but also of business needs. The narrowly educated network designer may fail to anticipate the social trend toward telecommuting, which will in turn increase the remote access bandwidth requirements. Similarly, the technically trained database designer may fail to include remote replication for a city susceptible to catastrophes.

Consequently, a course is needed which teaches the social and technical aspects of information systems through design activity. The ACM/AIS curriculum for Information Systems outlined such a course,[1] calling it Integrating Information Systems Technologies [1]. This paper describes the implementation of a course under this name.

Manuscript received December 16, 2004; revised January 22, 2006. This work was sponsored in part by the National Science Foundation under Grant ITR-0326309.

The author is with the Center for Decision Technologies, Wesley J. Howe School of Technology Management, Stevens Institute of Technology, Hoboken, NJ 07030 USA (e-mail: jnickerson@stevens.edu).

Digital Object Identifier 10.1109/TE.2006.873966

[1]The Association for Computing Machinery (ACM), and the Association for Information Systems (AIS) jointly publish a graduate curriculum for Information Systems.

The course accomplishes several educational goals. First, students learn to design complex integrated information systems. In addition, students learn the concepts underlying technical and social mechanisms of integration. Also, students improve their presentation skills. Last, students learn how information systems design is practiced as a profession.

This paper contributes to the field of design research, an area of growing interest to the computing disciplines. For example, the National Science Foundation has recently sponsored design workshops and programs [2], [3]. Information systems researchers have also shown an increased interest in design [4].

The paper proceeds in the following way. The next section reviews previous work in the teaching of design. The third section describes both the in-class teaching techniques and the course content. The fourth section presents the instructors' observations, and the fifth section analyzes the students' feedback.

II. Past Work on Design Teaching

The fields with the longest history of design pedagogy are architecture and engineering. While architectural design had traditionally been taught by the apprentice method, in the 19th century the Ecole des Beaux Arts institutionalized the teaching of architecture. Similarly, the Ecole Polytechnique institutionalized engineering education [5]. In the 1950s systems thinking began to influence design education [6]. In contrast to previous intuitive approaches to design, the design school at Ulm applied systems principles [7]. The author was taught by Horst Rittel, who brought many of the Ulm-based approaches to the University of California at Berkeley [8], [9].

In the management field, Simon called for a science of design [10]. Influenced by Simon and by the Ulm-based approaches, researchers have continued to invent systematic approaches to design [11].

Design in architecture schools is taught in the following way. In response to an assigned homework problem, students prepare a design in the form of a poster. The students bring the posters into class and put the posters on the wall. Then the professor leads a class discussion on a selection of the posters. This process, called a *crit* (short for critique), is repeated week after week.

Engineering schools have recently been paying increased attention to design. The undesired gap between theoretical teaching and the demands of the profession has led to *capstone* courses in design, as a way of integrating learning and preparing students for the environments they will face [12]. Such courses usually focus on a single problem, solved by a team. The success of these programs has led some schools to add what they call *cornerstone* design courses, which are taught

at the beginning of the program. Although design permeates many different courses in engineering, the design courses focus student attention on design processes rather than content mastery [12], [13]. In addition, in many fields of engineering, designs can be parameterized, and computer-aided design tools have been developed to allow for the immediate manipulation of these parameters. Engineering design courses often utilize these tools.

Information systems design has clearly been influenced by both engineering and architectural approaches. Engineering approaches are admired, but tools for information systems design are not as evolved, and even the most popular computer-assisted software engineering tools have low adoption rates for a variety of technical and social reasons [14]. Thus, instructors find themselves challenged in creating an environment in which to teach information systems design. Undergraduate Information Systems (IS) programs have sometimes introduced capstone design courses, which follow the architectural and engineering tradition of assigning a sequence of problems [15]. However, to the author's knowledge, information systems design has not been taught before using the crit method popular in architectural education, and this paper may be seen as an illustrative case of how the method can be used.

III. The Course

A. Student Audience

The course is being taught as a required course in a master's degree program in Information Systems at Stevens Institute of Technology. Approximately 12 sections a year are held, with about 20 students in each section. Most students are part-time working professionals. In general, these students work in the information technology departments of large corporations. Because of the location of the campus in New Jersey, neighboring Manhattan, students tend to be employed either by the financial industry or by the pharmaceutical industry. Also, many students come from other countries. Both full-time and part-time students have varied undergraduate backgrounds. Some students have technical degrees in computer engineering and computer science, while others have business or liberal arts degrees.

This heterogeneity of student backgrounds presents challenges for the instructor choosing course content. Such content needs to be technical enough to interest the engineers and computer scientists, but should be intelligible to the project manager with a business background.

B. Technical Content

Advances in software have created a large number of potential technologies related to integration. Databases, client/server, three-tier architectures, application servers, software buses, Web services, and other technologies have all been described as potential cures to the ills of fractured information systems.

The technical content of the course was selected with this range of technologies in mind, along with consideration of the computing disciplines spectrum. The ACM/AIS/IEEE curricula task force[2] describes this spectrum as a sequence of concerns running from the hardware, through the software, to the organization. The computing disciplines ordered in this way are: electrical engineering, computer engineering, computer science, software engineering, information technology, and information systems [16]. Thus, information systems can be characterized as the least technical, and the most social of the disciplines, influencing the nature of the material that is appropriate to teach in the course. For example, one of the few papers on pedagogy related to integration is Wegmann's description of a course in Enterprisewide IT systems [17]. Wegmann's course focuses on the software engineering section of the spectrum; therefore, it would not be appropriate for information systems students. Along the computing discipline spectrum, other courses which teach aspects of integration likewise have a different technical emphasis. In particular, systems engineering curricula often include a course on systems architecture; such courses tend to emphasize hardware over software and technical mechanisms over social mechanisms.

The concept map of Fig. 1 provides a visual layout of the two major foci of the course, the teaching of design skill and the communication of techniques for integration, both technical and social. This content is consistent with the ACM/AIS description of this course [1] and with the general skills associated with information technology and systems training in the ACM/AIS/IEEE curricula [16].

Two aspects of the design method are covered in depth—the divergent thinking associated with generating alternatives [18], and the convergent evaluation of these alternatives against many criteria. The integration techniques include both technical and social mechanisms for integration, and will be discussed in more detail after the following description of the teaching techniques.

C. In-Class Teaching Techniques

Class sessions are divided into formal lectures and critiques of student designs. The crit method from architecture, discussed in Section II, is applied to student designs in the following way. A design assignment is given, and all students return the next week to tape their posters on the walls of the classroom. The posters are required to be at least 17×22 inches in size. Once students post their own work, they look briefly at what the other students have created. To provide quick feedback to each other, the students are paired up. Each student presents the poster to his or her colleague, receives feedback from the colleague, and then listens, in turn, while the colleague presents. The process takes about fifteen minutes. In this manner, each student receives peer review every class. Not only do the students have an opportunity to practice their presentations, but also they teach each other, consistent with theories of sensemaking in education [19].

Next, a formal crit is performed. Several students who have volunteered during the previous class present their work to the entire class. After the presentation, the students and professor offer suggestions and opinions about the work, and compare it to other posters in the room. Technical and organizational issues are thereby debated in the context of a proposed solution. In the discussions the instructor serves as a guide, using the posters as platforms for introducing increasingly sophisticated concepts.

Four individual assignments are given over the first ten weeks of the class; each assignment is discussed twice in class, so that

[2]ACM, AIS, and IEEE jointly publish a set of curricula that span the computing disciplines.

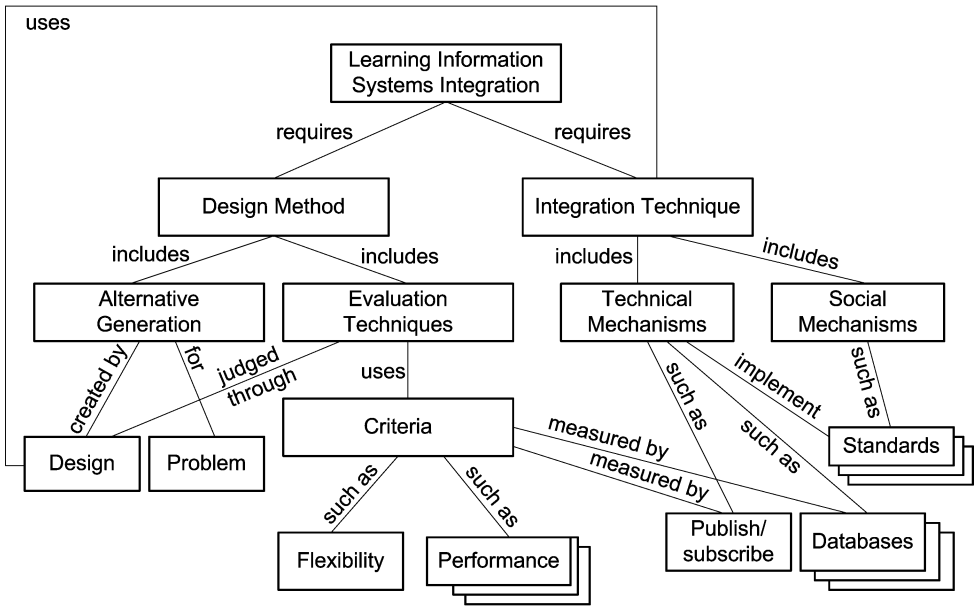

Fig. 1. Partial concept map for the course content.

students learn how to revise their designs in response to peer reviews. Educational literature suggests that students will not transfer knowledge in one domain to another unless they master the ability to abstract problems and the ability to think about the process of solving problems [20]–[22]. Both of these skills are developed through the discussions during the crits. In addition, the similarities between different types of systems problems are discussed explicitly in class following each assignment.

Final grading of the assignments is reserved until after the last poster is complete, when students are asked to submit a portfolio of the four completed designs for comments. This approach is consistent with educational research which finds that students learn more if given chances to revise their work prior to evaluation [23]. As the course progresses, the instructor provides verbal feedback immediately after the class to students producing substandard work.

The use of paper posters may seem anachronistic in an age of electronic slides. The paper posters, however, are an effective response to the constraints of the classroom. The instructors wish to expose students to many alternative designs for the same problem and would like students to be able to see and discuss all these designs at the same time. The instructors also wish to be able to see all the work at the same time so they can make comparisons. For both these reasons, some way of displaying 20 designs at the same time is needed.

One can imagine classrooms of the future in which projection walls might provide sufficient resolution to display many images simultaneously, eliminating the need to produce paper posters. Then, designs could be brought in electronically, with the added advantage that they could be manipulated in real time to suggest revisions. However, approximately 100 megapixels are necessary to display 20 posters (assuming $17'' \times 22''$ posters at 120 pixels per linear inch). Clearly, the classroom display of that much information through computer projection is at least several years away.

For the time being, then, physical posters present a cost-effective method for presenting large amounts of simultaneous visual information. However, since the posters will need to be revised in response to peer review, students should produce the posters using software, so that they can refine and regenerate their designs. Some commonly available diagramming packages make it simple to produce posters by breaking up large designs into tiled and slightly overlapping small pages that can be taped together without undue effort. This technique is how all the students produce their assignments.

The poster content has mandatory elements. Students are requested to provide both problem and solution scenarios, consistent with the ideas of Carroll [24]. Spatial and temporal diagrams are also requested, in the form of Unified Modeling Language (UML) deployment and sequence diagrams. In addition, students are asked to bring in their early designs in order to show the evolution of their ideas.

D. The Assignments and Course Chronology

Designs of information systems are often evaluated according to the criteria of functionality, reliability, performance, cost, maintenance, flexibility, security, and organizational impact. Some design problems naturally suggest particular criteria. The instructor chooses a range of problems that will prompt discussions of all of these criteria. For example, a problem statement for the redesign of the New York Stock Exchange might emphasize reliability and performance criteria, while a sales force automation problem might stress flexibility.

The sequence of assignments and the lectures are shown in Table I. The first lecture introduces the students to the syllabus. Also, a topical problem is picked for an in-class, team-based exercise; for example, a recent class was asked to design a hurricane response system. The students break into teams, draw a design, and report their ideas to the entire class. This first exercise gives the instructor a fast gauge of the background knowledge

TABLE I
THE COURSE CHRONOLOGY

#	Lecture topic	Design assignments due
1	Introduction	
2	Generating alternatives	Personal information system
3	Mechanisms of integration	Personal information system
4	Decision making	Database integration
5	Organizational theory	Database integration
6	Cross-enterprise integration	Notification system
7	Coordination theory	Notification system
8	Exam	Exam
9	Case study on resiliency	New Internet
10	Current integration standards	New Internet
11	Emerging standards	
12	Emerging technologies	
13	Summary lecture	
14	Student presentations	Group project

level of the students and gives students a safe way to practice new skills.

The first homework assignment, the design of a personal information system, is based on an early assignment given during the 1970s in Horst Rittel's Urban Planning course at UC Berkeley. Architecture students often responded with physical drawings of their living space. Today, information systems students often respond with designs relying solely on personal digital assistants. These information systems students make the same mistake the architecture students did. They focus on the static objects of the system, rather than on the processes that produce the information. After the posters are presented at the beginning of the second class, process-oriented ways of solving the problem are discussed [25], [26], and the students are asked to revise their posters for the next class.

The second lecture reviews the creation of scenarios. Scenario-based techniques are taught in most systems analysis courses, yet the students often need reinforcement. In particular, many do not grasp the importance of specificity in the scenario. Students also are unclear on the potential use of a scenario to envision not only the problem, but also a future solution. Three sources of scenario ideas are discussed: Jacobson [27], Schwartz [28], and Carroll [29]. Systems approaches to generating alternative designs are also introduced. These include Rittel's method of systematic doubt, in which each logical term in a scenario is negated as a way to generate potential solutions.

The third lecture presents mechanisms of integration, including database integration, *publish-and-subscribe*, and *store-and-forward*. General principles of synchronous versus asynchronous methods of integration are presented. *Enterprise application integration* is explained.

The next assignment comes directly from an industry problem the author has encountered in many corporations. Students are asked to design the integration of seven customer databases in a company, without changing the applications which rely on these databases. The problem is deeper than it first appears, and solving it calls for a thoughtful analysis of read and write transactions.

The fourth lecture discusses decision making as it relates to the evaluation of design. Design evaluation is a multi-attribute decision problem under conditions of uncertainty; students learn how such problems are described and handled. They also learn how engineering approaches can be used to find optimal solutions. Specifically, they learn the criteria used to evaluate systems and several ways of trading off these criteria [30]–[32]. Automated approaches to generating design are also noted [33]. The differences between parametric engineering design methods and the ad hoc methods of information systems are discussed, as are the social aspects of the evaluation process [34].

The fifth lecture discusses organization theory in relation to integration [35]. Recent research into social networks is also explained because this research shows how integrating technologies such as e-mail are used in practice [36].

The next design problem asks students to design a notification system for a large news organization and to estimate the cost of their solution. Students are expected to select the appropriate integration mechanism for this problem, such as *publish-and-subscribe*. Imaginative students sometimes design less costly solutions based on social networking.

The sixth lecture discusses cross-organizational integration, which is the expansion of companies' information systems to include the systems of their customers and suppliers [37]. The seventh lecture looks at coordination science [38], [39] which provides a way of analyzing and designing business processes. Alexander's concept of a design pattern is explained [40], [41]. Also discussed are architecture frameworks [42] and their use in producing an architecture document [43].

The eighth class is an exam. Short questions about the course content and the readings are followed by a long design question. For example, students might be asked to design a sales force automation system.

A final individual problem is also assigned. Students are asked to redesign the Internet protocols in the face of increased mobility and increased commercial usage; Joy's ideas on heterogeneity in software are discussed to stimulate conversation [44], as is the Turing award lecture of Cerf and Kahn [45]. The assignment helps integrate the material of the course and leads to a deeper understanding of the infrastructure that increasingly underlies large integrated systems.

The ninth lecture reviews the results of the exam and then turns to addressing the resiliency of systems in detail, using a case study approach. Ideas of distribution and replication are discussed.

The tenth lecture focuses on software integration standards. The design of standards is discussed, because it provides an example of how complex technical design can be accomplished in groups. The eleventh lecture discusses areas where coordination standards are still being developed, such as the Semantic Web and Web services choreography [46]. Students learn why some companies decide to adopt such emerging standards.

The group project is assigned. This assignment may vary according to the current needs of the surrounding industries. For example, a recent assignment asked students to design disaster plans in anticipation of a flu epidemic. The last assignment is produced as both a poster and a written report detailing the architecture. The report provides students an opportunity to show the depth of their thinking.

The twelfth lecture discusses emerging technologies; recently, this lecture has focused on mobile ad hoc networks. The thirteenth lecture is a summary lecture, and the fourteenth class consists of the group presentations of the final assignment.

IV. Instructor Experiences

The author and six other instructors have now taught this course using the curriculum described in this paper. All agree that the experience is very different from teaching using more traditional techniques.

The crits change the classroom environment. When assignments are submitted as written reports, the viewing of the work is private, between student and professor. In the crit environment, however, everyone sees each other's work.

The instructors believe the students' ability to present ideas, answer criticism, and ask questions increases more rapidly in this open environment; one wrote that the poster method "makes the material much more alive—you can get more classroom discussion than with other methods." In contrast to slide presentations given one-by-one, the method allows for what one instructor calls "cross-poster comparison." Also, since the "posters stay up," he can "refer to [student's] solutions during the lecture period."

The course presents challenges: because the technique is new to them, "students will be apprehensive." Shy students "may be intimidated" by bolder students' work in the first classes. The first few classes in general will not produce many great posters. Some instructors find showing previous students' work, good and bad, is important so that students get a sense of what is possible.

The consensus of the instructors is that the early classes are the most delicate to manage; the goal is to encourage the good design work without discouraging the less experienced students. One instructor says "in-class exercises are crucial" because students get quick feedback from their peers and the instructor before undertaking the homework assignments.

Another instructor noted that the posters can be used "to socialize the student into the discipline." Some instructors think the poster approach helps students "get directly to the point"; however, "brevity of expression may also hide information."

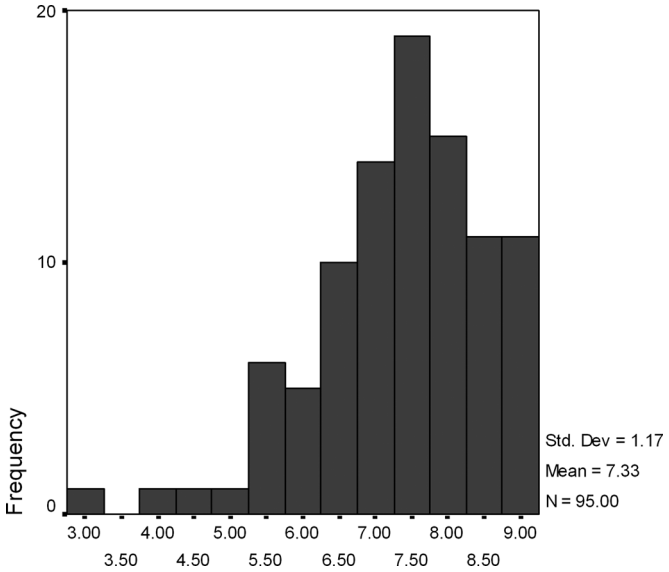

Fig. 2. A histogram of the mean assessment for the assignments from the 95 students. Nine means *extremely useful*.

V. Student Assessments

The instructors ask students for anonymous feedback at the end of every semester by presenting a student feedback form. This form asks the students to assess the usefulness of each of the assignments. This form also asks them what they think they have learned. All students attending the last class of the semester filled out the form.

Fig. 2 is a histogram of 95 responses to this questionnaire, representing nearly all students in five sections of the course the author has taught over a period of three semesters. The students were asked to rank each assignment on a scale from 1 to 9, *not at all useful* to *extremely useful*.

The mean response was 7.3. The group assignment had an average value of 8.2, with a standard deviation of 1.2; the first assignment had an average value of 6.5, and a standard deviation of 2. The last individual assignment had an average of 8.0, and a standard deviation of 1.5.

Why was the initial assignment seen as less useful than later assignments? Students noted that they did not know what to do the first time. They also did not know what to expect in the critique of the first assignment. Thus, the students might have been remembering their feelings of confusion when they made their evaluations.

Just as the students' work improved throughout the semester, their perception of the usefulness of the assignments improved. The last individual assignment and the group assignment which takes place at the end of the course were both rated highly. This result held up even when new assignments were substituted for old, leading to the tentative conclusion that as students' mastery increases, their perception of the value of the assignments also increases.

The author was interested in finding out what students think of the poster technique since they have not encountered this technique before in the program. One question reads "overall, how effective did you think the posters were in teaching application of course concepts compared to traditional assignments?"

The students were asked to answer either a) *more effective*, b) *about the same*, or c) *less effective*. Two percent left the question blank; 6% said the technique was less effective; 16% said the technique was equally effective; and 76% found the posters more effective than traditional assignments.

Students were asked "do you think the course has improved your ability to design systems?" Four percent did not answer the question; 5% said no; 7% hedged their answer (for example, "somewhat"); and 84% said yes.

Students were also asked open-ended questions on the advantages and disadvantages of the poster approach.

In discussing the disadvantages, 24 students noted that the assignments took longer than normal assignments. A few complained about the logistics of printing and assembling the posters. Five students thought that visualizing ideas was a difficult task for which they were not prepared ("sometimes it's hard to draw your thoughts on a diagram"), and several called for tutorials on the creation of the appropriate diagrams. One, and only one, student noted that "not everyone is creative."

Even with these disadvantages, the students generally liked the poster approach. In their discussion of advantages, 23 said they liked the visual, configurational, all-at-once approach of the poster. In the words of one student,

> "Presenting a hyperlinked concept in a 2-dimensional space is extremely useful to navigate back and forth as opposed to serialized powerpoint presentation. Additionally, it gives a holistic view of the entire design."

Nineteen found that being able to see what other students were doing was valuable. They observed that the posters were "quick to understand." Fifteen liked the creativity they were called on to exhibit through the generation of alternative designs: "thinking out of the box is well practiced in the course." Thirteen thought the poster format "forces you to be concise" and improved critical thinking. They liked presenting ideas spatially "instead of writing essays" and liked the ability to "see the thought patterns of other people."

VI. Conclusion

The complexity of modern information systems creates a problem for instructors. On the one hand, specialized knowledge needs to be taught, and this knowledge is organized into technology layers. On the other hand, real world problems range across these layers. The author has presented the curriculum for a course which, through a series of design exercises, teaches students how to think about complex and realistic problems. The content of the course is drawn from industry practice and includes an analysis of different mechanisms for integration. The design exercises use a poster technique which is common in design education but rarely used in information systems education. Students think the approach works better than traditional techniques and find the overall course valuable.

Acknowledgment

The author wishes to acknowledge the encouragement to develop the course from J. N. Luftman, E. A. Stohr, and S. Tewksbury. The author appreciates the suggestions of the other course instructors, K. Bent, R. Kelly, M. McDowell, M. zur Muehlen, A. Saltzman, and W. Wong, as well as the contributions of many students. In addition, S. Desai, E. Friedman, A. Kay, and R. E. Watson provided helpful insights.

References

[1] J. T. Gorgone and P. Gray, Eds., (1999) MSIS 2000: Model Curriculum and Guidelines for Graduate Degree Programs in Information Systems: Report of the Joint ACM/AIS task Force on Graduate IS Curriculum. [Online]. Available: http://www.aisnet.org/Curriculum/msis2000.pdf

[2] K. Sullivan. (2003) Preliminary Report: NSF Workshop on the Science of Design: Software and Software-Intensive Systems. [Online]. Available: http://www.cs.virginia.edu/~sullivan/sdsis/SDSIS%20Preliminary%20Report%20 020 210b.pdf

[3] R. Ramakrishnan, P. Bernstein, and A. Halevy. (2003) Workshop on the Science of Design for Information Systems. [Online]. Available: http://www.cs.wisc.edu/sdis03/

[4] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information research," *MIS Quart.*, vol. 28, no. 1, pp. 75–105, 2004.

[5] I. Grattan-Guinness, "The ecole polytechnique, 1794–1850: differences over educational purpose and teaching practice," *Amer. Math. Month.*, Mar. 2005.

[6] L. Bertalanffy, "An outline of general systems theory," *Brit. J. Phil. Sci.*, vol. 1, no. 2, pp. 139–164, 1950.

[7] R. Spitz, *Hfg Ulm: the View Behind the Foreground: the Political History of the Ulm School of Design, 1953–1968*. Stuttgart, Germany: Edition Axel Menges, 2002.

[8] H. W. J. Rittel and M. M. Webber, "Dilemmas in a general theory of planning," *Policy Sciences*, vol. 4, no. 2, pp. 155–169, 1973.

[9] H. W. J. Rittel and University of California Berkeley, "APIS, A Concept for an Argumentative Planning Information System," Institute of Urban & Regional Development, University of California, Berkeley, Working Paper no. 324, 1980.

[10] H. A. Simon, *The Sciences of the Artificial*. Cambridge, MA: MIT Press, 1969.

[11] M. Nadin, "Computational design. Design in the age of a knowledge society," *Formdiskurs, J. Design Design Theory*, vol. 2, no. 1, pp. 40–60, 1996.

[12] C. L. Dym, A. M. Agogino, O. Eris, D. D. Frey, and L. J. Leifer, "Engineering design thinking, teaching, and learning," *J. Eng. Educ.*, vol. 94, no. 1, pp. 103–120, Jan. 2005.

[13] G. Pahl and W. Beitz, *Engineering Design: A Systematic Approach*. New York: Springer, 1996.

[14] W. J. Orlikowski, "Case tools as organizational-change—Investigating incremental and radical changes in systems-development," *MIS Quart.*, vol. 17, no. 3, pp. 309–340, 1993.

[15] K. Surendran, I. C. Ehie, and C. Somarajan, "Enhancing student learning across disciplines: A case example using a systems analysis and design course for MIS and ACS majors," *J. Inf. Technol. Educ.*, vol. 4, pp. 257–274, 2005.

[16] Joint Task Force for Computing Curricula. (2004) Computing Curricula 2004. [Online]. Available: http://www.acm.org/education/Overview_Draft_11-22-04.pdf

[17] A. Wegmann, "Designing enterprisewide IT systems," *IEEE Trans. Educ.*, vol. 47, no. 4, pp. 490–497, 2004.

[18] J. Baer, *Creativity and Divergent Thinking: A Task-Specific Approach*. Hillsdale, NJ: L. Erlbaum Associates, 1993.

[19] R. D. Pea, "Learning scientific concepts through material and social activities: conversational analysis meets conceptual change," *Educ. Psych.*, vol. 28, no. 3, pp. 265–277, 1993.

[20] G. Salomon and D. N. Perkins, "Rocky roads to transfer: rethinking mechanisms of a neglected phenomenon," *Educ. Psych.*, vol. 24, no. 2, pp. 113–142, 1989.

[21] M. L. Gick and K. J. Holyoak, "The cognitive basis of knowledge transfer," in *Transfer of Learning: Contemporary Research and Applications*, S. M. Cornier and J. D. Hagman, Eds. New York: Academic, 1987, pp. 9–46.

[22] J. G. Greeno, D. R. Smith, and J. L. Moore, *Transfer on Trial: Intelligence, Cognition, and Instruction*. Norwood, MA: Ablex, 1993.

[23] K. Bain, *What the Best College Teachers Do*. Cambridge, MA: Harvard University Press, 2004.

[24] J. M. Carroll, *Scenario-Based Design: Envisioning Work and Technology in System Development*. New York: Wiley, 1995.

[25] D. Allen, *Getting Things Done: the Art of Stress-Free Productivity*. New York: Viking, 2001.

[26] S. Fertig, E. Freeman, and D. Gelernter, "Lifestreams: an alternative to the desktop metaphor," in *Proc. CHI*, 1996, pp. 410–411.
[27] I. Jacobson, *Object-Oriented Software Engineering: A Use Case Driven Approach*. Reading, MA: Addison-Wesley, 1992.
[28] P. Schwartz, *The Art of the Long View*, 1st ed. New York: Doubleday/Currency, 1991.
[29] J. M. Carroll, *Making Use: Scenario-Based Design of Human-Computer Interactions*. Cambridge, MA: MIT Press, 2000.
[30] R. L. Keeney and H. Raiffa, *Decisions With Multiple Objectives: Preferences and Value Tradeoffs*. Cambridge, U.K.: Cambridge University Press, 1993.
[31] D. P. Grant, *Design by Objectives*. San Luis Obispo, CA: Design Methods Group, 1982.
[32] Y. Hatamura, *Decision-Making in Engineering Design*. New York: Springer, 2005.
[33] E. K. Antonsson and J. Cagan, *Formal Engineering Design Synthesis*. New York: Cambridge University Press, 2001.
[34] M. Bergman, J. King, and K. Lyytinen, "Large scale requirements analysis revisited: the need for understanding the political ecology of requirements engineering," *Req. Eng. J.*, vol. 7, no. 3, pp. 152–171, 2002.
[35] P. R. Lawrence and J. W. Lorsch, *Organization and Environment; Managing Differentiation and Integration*. Boston, MA: Division of Research, Graduate School of Business Administration, Harvard University, 1967.
[36] P. S. Dodds, D. J. Watts, and C. F. Sabel, "Information exchange and the robustness of organizational networks," *PNAS*, vol. 100, no. 21, pp. 12 516–12 521, 2003.
[37] E. A. Stohr and J. V. Nickerson, "Intra Enterprise Integration," in *Competing in the Information Age: Align in the Sand*, 2nd ed, J. Luftman, Ed. New York: Oxford University Press, 2002, pp. 227–251.
[38] K. Crowston, "A taxonomy of organizational dependencies and coordination mechanisms," in *Tools for Organizing Business Knowledge: The MIT Process Handbook*, T. W. Malone, K. Crowston, and G. Herman, Eds. Cambridge, MA: MIT Press, 1994, pp. 85–108.
[39] T. W. Malone, K. Crowston, and G. A. Herman, *Organizing Business Knowledge: the MIT Process Handbook*. Cambridge, MA: MIT Press, 2003.
[40] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press, 1977.
[41] E. Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.
[42] P. Kruchten, "Architectural Blueprints—The 4 + 1 view model of software architecture," *IEEE Software*, vol. 12, no. 6, pp. 42–50, 1995.
[43] M. Maier and E. Rechtin, *The Art of Systems Architecting*, 2nd ed. Boca Raton, FL: CRC Press, 2000.
[44] B. Schlender, "Joy after sun," *Fortune*, Sep. 29, 2003.
[45] V. G. Cerf and R. E. Kahn, *ACM Turing Lecture: Assessing the Internet: Lessons Learned, Strategies for Evolution, and Future Possibilities* Philadelphia, PA, 2005.
[46] M. zur Muehlen, J. V. Nickerson, and K. D. Swenson, "Developing web services choreography standards—the case of REST vs. SOAP," *Dec. Supp. Syst.*, vol. 40, no. 1, pp. 9–29, 2005.

Jeffrey V. Nickerson (S'86–M'87) received the Ph.D. and M.S. degrees in computer science from New York University, the M.F.A. degree in graphic design from Rhode Island School of Design, and the B.A. degree in visual design from the University of California at Berkeley.

He is an Associate Professor at Stevens Institute of Technology, Hoboken, NJ, in the Wesley J. Howe School of Technology Management. He is the Director of the Center for Decision Technologies. His research interests include mobile ad hoc networks, sensor networks, standard making, and design. Prior to joining Stevens Institute of Technology, he was a partner at PriceWaterhouseCoopers, consulting on issues related to system design.