Teaching IS Design and Development in a Group Learning Setting

Gail L. Rein

Department of Management Science and Systems
State University of New York

Abstract

This paper describes the instructional approaches for an eight-week, team-based computing project used to teach the principles of information systems design and development to all undergraduate business students at the State University of New York at Buffalo. Three aspects of the project relevant to group learning environments are discussed:

- the process used to form skill-balanced teams,
- the use of scenarios to elicit system requirements (a nontraditional approach), and
- the use of an iterative, six-phase rapid proto-typing process.

The paper concludes with a description of the collaboration technology to be developed to support the prototyping process and instructional approaches described in the paper.

Keywords — cooperative learning, project-based learning, computer-supported collaborative learning, educational groupware, scenario-based requirements, rapid prototyping process.

1. Introduction

A common attitude among information systems (IS) educators is that we should teach undergraduates conceptual skills[1] and leave the learning of practical, or technical, skills[2] to the workplace [15]. This attitude creates a near-crisis situation: the majority of business graduates enter the workplace without the most basic technical skills they need for their jobs.

The ever-increasing demand for graduates with technical IS skills is a direct consequence of pressure from three trends. First, the shortage of computer science graduates [1] [5] means that businesses are being forced to hire more business and liberal arts graduates for their entry-level positions [13]. Second, the ubiquitous application of information technology in every aspect of business operations [1] [22] means that even students who are not IS majors may at some point in their careers work on a corporate re-engineering project, be involved in decision-making activities associated with the building of a strategic informatio 16pate in the definition of an enterprise-wide computing solution. Third, the proliferation of fourth-generation programming environments means that more firms are using prototyping techniques to develop their IS applications [6], and these environments can be used successfully by non-programmers.

How can we better prepare business graduates for these workplace demands? The School of Management at the University of Buffalo (UB), like most business schools, requires all its undergraduate business students take an introductory course in information systems. For most students (all of the non-IS majors), it is the only IS course they take. The UB course is a technical,

[1]Conceptual skills address organizational issues, such as the overall organizational design of information systems, the relationship of the IS function with other organizational functions, and the organizational impact of informations systems.

[2]Practical skills address issues in the management of systems development, such as feasibility and justification, the systems life cycle process, information requirements, prototyping, integration of systems, restructuring of existing systems, project management, configuration management, and systems implementation.

hands-on course designed to develop both conceptual and practical IS skills. In the course, students learn how to use fourth-generation software packages for solving business problems. They also learn about and use two electronic communication tools: Usenet news and email. Required work for the course includes two spreadsheet assignments, two relational database assignments, and a team computing project.

The focus of this paper is on the computing project, as it provides a group learning context for developing students' team skills and their systems design and development skills. The project is authentic and demanding in that it requires multiple complementary skills not commonly found in a single individual. Students with differing strengths and weaknesses must learn how to work together in order to produce a satisfactory product.

Three aspects of the project relevant to group learning environments are discussed in this paper:

- the process used to form skill-balanced teams,
- the use of scenarios to elicit system requirements (a nontraditional approach), and
- the use of an iterative, six-phase rapid proto-typing process.

We conclude with a description of the collaboration technology that we plan to develop to support the prototyping process and instructional approaches described in the paper.

2. Team Computing Project Overview

The computing project accounts for 40% of the typical student's total effort for the course. Team members must develop a prototype information system using a database management system; write a term paper describing their prototype; and give a 20-minute presentation, including a live demonstration of their prototype, to the class.

Ideas for the project are proposed by the students and subject to approval by the teacher. Examples of systems that have been prototyped in our classes over the past two years include a system for tracking customers' print orders as their film goes through the photo-finishing process, a home-video library manager, a system for selecting seats and purchasing tickets for cultural events, an airline baggage tracking system, a perpetual inventory system for a golf store, an apartment locator, a system for scheduling swimming pool service calls, a system to expedite emergency room check-ins, a system for reuniting lost pets with their owners, a feed management system for a race horse stable, a system for tracking fitness club memberships, a football player recruitment system, and a car pool system for matching drivers and riders.

The project engages students in intensive investigation efforts, qualifying it as a project-based learning experience according to the criteria set forth by Blumenfeld, Soloway, et al. [2]. Students are motivated because they pick an application that personally interests them and that addresses a problem they have experienced in their lives. They pursue solutions to nontrivial problems by debating ideas, asking questions, designing plans of attack, observing and analyzing the outcomes of their prototyping experiments, and communicating their ideas and findings to others. Particularly important for motivation, their efforts ultimately lead to the creation of the three tangible (and significant) artifacts: a *working* prototype, a paper, and a presentation.

Another important motivational consideration is grading. We use criterion-referenced, or mastery, grading methods [8], which we have extended to team projects [18]. Criterion-referenced grading, which uses grades to measure students' level of mastery of the subject material against a stated standard rather than to rank them within the class, has profoundly altered the attitudes of both students and teacher so that learning has become what Koschmann [11] calls an 'active process.'

3. Team Formation Process

In the sixth week of the course, after important introductory material has been covered, one class meeting is devoted to an hour-long team skills exercise in which the students sort themselves into teams of five (our classes have 60 to 70 students). The exercise helps the students form skill-balanced teams so that each team has members with the five skills important for successfully undertaking the project: programming, speaking, writing, research, and process skills.[1]

The exercise starts with a teacher-guided discussion of the goal of the exercise, which is for each person to join a high-performance team.

> "Being a member of a high-performance team is one of the most exhilarating experiences you can have. The founding fathers who wrote the U.S. Constitution... the team of scientists who put the first astronauts on the moon... the computer design team in Tracy Kidder's *The Soul of a New Machine*... all of these examples are high-performance teams that we continue to talk about and admire. One distinguishing attribute of a high-performance team is that it *consistently* achieves outstanding results... what are some of the other qualities of a high-performance team?"

[1] Instead of letting the students pick their team members, we have also assigned teams using skill data from a questionnaire that asks each student to rate him/herself on the set of team skills important for the project. See [19] for details.

The class usually chooses a sports team to think about the attributes that distinguish it as a high-performance team. As the students identify attributes, they are written on the board where everyone can see them. The importance of complementary skills always comes out in the discussion, and this provides the segue into the next step of the exercise.

We ask all of the students who feel that leadership is one of their skill strengths to stand. This is a safe, unintimidating way to begin the team formation process, because about half of our students rate themselves as having leadership skills.[1] We talk about the role of the leader in a high-performance team and the fact that it is often a rotating responsibility [10] [19], but "just to begin the team formation process today, we will seed each team with a strong leader." The people who are standing are asked to decide quickly among themselves which of them will start forming teams (the number of teams depends on class size). At this point, one-fifth of the class, slightly fewer than half of the leaders, are left standing; the others sit down.

The remaining students are instructed to organize themselves into five equal-sized skill groups in designated parts of the room. The skill groups are Speakers, Programmers, Writers, Researchers, and Process Experts. The students quickly sort themselves into the groups, but rarely are the groups the same size (usually there is a shortage of Programmers and Speakers). Many students have more than one skill strength, so simply reminding them of this and encouraging the multi-skilled students to move to an undersized group is all that is needed to get the desired balance.

Next, the leaders visit the groups to interview people for their teams. The leader is reminded to consider his/her own skill strengths (other than leadership) in forming the team. We encourage all the members of the growing team to participate in the interview process.

At the end of the exercise, the students have organized themselves into 12 to 14 teams of five such that each team has a set of people with complementary skills important for completing the project. Perhaps even more importantly, the students have already learned a great deal about each other, and they spend the last 10-15 minutes of the class period in their teams discussing project ideas.

4. Prototype Development Process

The project requires eight weeks and uses a rapid prototyping process consisting of six phases (each phase will be described shortly). Figure 1 shows the project time line by course week. The teacher's approval is required before a team proceeds from one phase to the next.

For most students, the project is their first programming experience. The primary goal of the project is to give students an appreciation of (by doing) what is involved in the development of an information system, not to create an entire class of programmers. Not only do the students learn some basic system development skills, but they also are introduced to several IS design principles and methods. They have to learn the principles and methods well enough to apply them to their projects. The teacher must be willing, and able, to provide ongoing coaching to the teams.

The project is difficult for most students and we talk about why. System design is a 'wicked problem' [3] [20]. In contrast to 'tame problems,' wicked problems have no definitive formulation, no stopping rule (the projects are never really done; they can be refined and improved ad infinitum), solutions are not true-or-false (they are good-or-bad; we use terms like "good," "better," and "good enough" to evaluate them), and every one is essentially unique (no two projects are the same).

Wicked problems, because they have no definitive formulation, cannot be solved in a straightforward manner. It is therefore to be expected that a team will gain a better understanding of an earlier project phase when doing a later one. Teams are encouraged, and expected, to turn in revised versions of earlier phases whenever they feel the need to refine or redo them. Many of the teams in our classes redo their requirements and database designs two, three, and even four times. Figure 1 depicts this iterative property of the process: the bars for the requirements and design phases have refinement extensions, and each phase (bar) in the project overlaps at least one other phase (bar).

The teacher must give careful thought to the system software that the students use for developing their prototypes.[2] Most modern database management systems provide decent fourth-generation programming tools, which make these systems accessible to non-programmers. The selected system should provide a range of useful application development tools, such as report generators, data-entry screen generators, high-level menu management functions, and easy-to-use run-time debuggers.

[1]On average, 47% of the students in our undergraduate IS classes have self-assessed leadership skills. The rare skills are programming (13%) and speaking (20%). These statistics are the average of the class averages from four semesters of skill questionnaire data collected from the author's sections of the course.

[2]In a real systems development situation, the students might have to make this decision; but the limited time for the project and the background of the students precludes this.

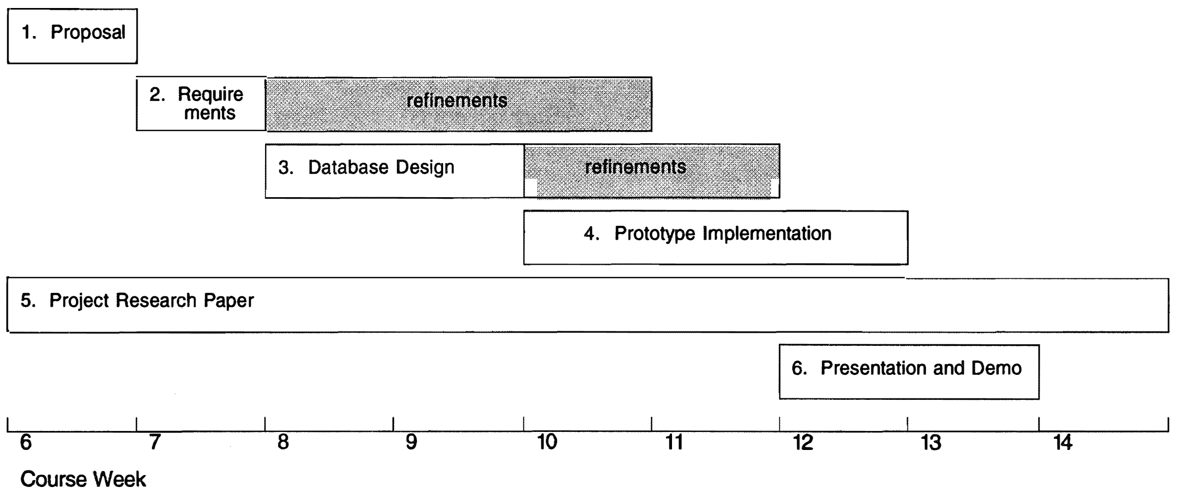

Figure 1. Project phases.

4.1. Phase 1 — Proposal

In Phase 1, each team writes a proposal, which is a brief statement (one or two paragraphs) of the problem or application to be considered and researched. Teams are instructed to focus on answering *why* and *what*, not how, questions. We suggest students start with a problem or frustration they have experienced (e.g., losing baggage in an airport) and then think about how an information system might resolve the problem. The best projects are those that are problem-driven, not technology-driven.

4.2. Phase 2 — Requirements

In Phase 2, the teams identify the functional requirements of their systems. They think about who would use the proposed systems, what information the systems should provide, and how the information might be used. In this project phase, the teams are instructed to focus on answering detailed questions about *what* their proposed systems will do, rather than how they will do it.

The deliverables from this phase are a scenario and a context-level data flow diagram (DFD). The scenario is a vignette that describes a real-world situation, both before and after the proposed information system exists; the DFD clarifies the system's boundaries and how it interacts, from a data perspective, with the end-users of the system. Students are instructed to write the scenario before attempting the DFD.

4.3. Phase 3 — Database Design

In Phase 3, each team creates a database with at least two relations and populates it with a combined total of at least 50 records. A printout of the database and supporting documentation is submitted for teacher feedback. The supporting documentation contains descriptions of each of the fields in each relation, the team's rationale for its database design decisions, and "proof" that its relations are in third normal form. Frequently, a revised or refined scenario and DFD are turned in with Phase 3.

4.4. Phase 4 — Prototype Implementation

In Phase 4, the teams implement the functions described in their scenarios. To meet the project requirements, their implementations must have a minimum of five well-commented procedures. They turn in diskettes of their prototype systems and printouts of all data files and programs. We also ask the teams to turn in current iterations of Phases 2 and 3 and any other supporting material with their project diskettes.

4.5. Phase 5 — Paper

In Phase 5, each team prepares a final report that includes everything from Phases 1 through 4 (in a refined, final form) and additional relevant material from its research. A detailed handout describing the required sections of the paper is given to the students.

4.6. Phase 6 — Presentation and Demonstration

In Phase 6, each team gives a 20-minute presentation of its project to the class. Each presentation includes a discussion of the technical challenges the team encountered in developing its prototype and a live demonstration of the prototype.[1]

The room is charged with emotions on presentation days. The students presenting are nervous and excited; the rest of the class is attentive, interested, supportive,

[1]Our classes are taught in a modern, multi-media equipped classroom. The prototypes, running on a 486 computer housed in the lecturer's podium, are projected onto a large public screen.

and encouraging. As one student put it, "This computing project is a really intense project. At the end, it's fun to show off something you've worked hard on; and it's interesting to see what the other teams have produced." The presentations are the high point of the project, and most of the teams are rightfully proud of their prototypes.

5. Central Role of the Scenario[1]

The scenario, first generated in Phase 2 and refined in Phases 3 through 6, is at the very core of the prototyping process used for the computing project.

At first (Phase 2), most students are reluctant to write a scenario. This reluctance is not surprising: requirements analysis is one of the most difficult activities in information systems development [4] [23]. It is difficult for people to conceptualize and then describe their information needs in terms of system functions.

It is critically important that the teacher encourage students to be explicit. The scenario must describe actual events, including people, their dialogs, and even numbers, where appropriate. The more explicit the scenario is, the more it helps the team focus on a useful, demonstrable subset of functions for its prototype. Without an explicit scenario, members of the team attempt to implement more than they can manage—they can easily end up with a prototype that tries to do everything, but that does nothing well. The team should not be allowed to move to Phase 3 until an adequately explicit scenario has been written.

Once explicit, the scenario gives the members of the team a concrete mental image of what they are building. This image guides them in their database design decisions (Phase 3) and helps them focus their implementation efforts (Phase 4). As work progresses through both these phases, the scenario is continually refined and made more and more explicit.

Most of the teams dramatize their scenarios to demonstrate their prototypes (Phase 6). The pressure of having to demonstrate prototypes makes many teams add significant final refinements to their scenarios. Many creative and entertaining scenarios have been presented, most of them making the prototypes appear to do more than they really do.

The scenario serves an important new purpose at the final stage of the project: it creates powerful mental images of the possibilities suggested by the prototype in the minds of the viewers. In the classroom setting, the viewers are other students in the class; but in the workplace, the viewers would be end-users or project sponsors.

[1]For an example of a scenario, please write the author.

6. Collaboration Technology

Currently, the computer in the UB computing project plays a 'passive role' [16] with respect to the collaborative learning experience. The database management system (DBMS) software that teams use to develop their prototypes is the primary computer-based component in the project. This software is designed for individual use—it does not have any special properties that allow it to support the collaborative learning process.

The learning process is managed by the teacher through interactions with the teams. The teams' progress is tracked by means of an elaborate spreadsheet, using labor-intensive manual procedures to record activity events. It is also necessary to maintain files of photocopies of the deliverables from each stage for each team—these files are, in essence, the only project repositories to which the teacher has assess (for the teams, their diskettes are repositories).

We are in the early stages of designing a two-layered, computer-based, collaborative system to support both the prototyping process and the instructional approaches described in this paper. The inner layer will support each team in the prototype development process. The outer layer will support the teacher in coaching the teams and tracking their progress.

For the inner layer, we plan to build a shell around the DBMS that will provide *activity-level coordination*[2] functions to guide each team through the six-phase rapid prototyping development process. Since a team may be (and usually is) working on more than one phase at a time, the shell must support *multiple endeavors*. We also have plans for developing a *single-stage* groupware editor that will support multiple people simultaneously writing a scenario. The editor will provide *object-level coordination* of the objects (e.g., the characters or roles, their scripts, and the functional requirements) that make up explicit scenarios.

For the outer layer, we plan to build an *activity-level coordination* system, based on electronic mail, which will allow the teacher to receive deliverables from the teams and return them with comments. This layer will be more complex in that it must ultimately provide *total inspection* capabilities, allowing the teacher to observe the progress of the teams' projects (i.e., the *endeavors*) in the inner layer. Eventually, this system should provide *second-order inspection* capabilities, reporting on statistics of interest such as the average time it takes the teams to complete each phase and the average number of iterations for a particular task or phase.

In summary, we note that the two-layer system described above provides support for all forms of commu-

[2]Italics in this and the next paragraph are concepts defined in Ellis' and Wainer's "Conceptual Model of Groupware" [7].

nication *and coordination*[1] that characterize 'active learning processes' [11] [12]: 'peer-to-peer,' 'student-to-teacher,' and 'teacher-to-student' [17]. Specifically, the inner layer of the system provides a 'computer-mediated learning environment' [16] for peer-to-peer communication and coordination,[2] and the outer layer of the system supports teacher-to-student and student-to-teacher communication and coordination.[3]

7. Conclusion

Business information systems are developed in a variety of ways. They may be created by integrating purchased pieces, by building them from scratch, by modifying existing systems, or by a combination of all or some of these development strategies. It is important that business students learn practical problem solving approaches so they can more effectively tackle the problems they will face in their jobs. In particular, all business students (not just the IS majors) need the experience of building a system as team members [14] [22] using "real life" software tools [13].

In this paper, we have described an eight-week, team-based computing project used to teach the principles of information systems design and development to all undergraduate business students at the State University of New York at Buffalo (UB). Although it may be standard practice to teach IS design and development principles using project-oriented courses that require a prototype, a paper, and presentation, the literature does not contain many papers describing these projects.[4]

The UB project requires that students work in teams to develop demonstrable, prototype information systems using a database management system similar to, or the same as, what businesses use to develop their applica tions.[5] The learning that happens is a 'constructive process' [21], and it develops students' practical and technical skills in many ways. For example, they learn system development tactics such as incremental coding techniques. They learn how to read technical manuals and apply what they read to solve real world problems. They even indirectly pick up technology evaluation skills: the experience of developing an application generates an awareness of what to look for when selecting hardware or software products. When they enter the workforce, the majority of our students may develop IS applications only for their own use and never for use by others (most of the students are not IS majors), but after completing this course they will have gained an invaluable appreciation for what IS can and cannot do and how complex it is to create a useful system.

Our future research will explore the technical issues of how to build the computer-based collaborative system described in Section 6, as well as many usage and process issues. With regard to latter, we believe that we are onto something exciting and promising with the scenario-based process approach. An empirical study by Zmud, Anthony, and Stair [23] provides modest evidence that mental imagery protocols may outperform traditional approaches for eliciting requirements in ill-structured task contexts, but more empirical work needs to be done. Above all, we hope this paper leads to future collaborations with behavioral and educational experts.

Acknowledgments

This paper benefited from the constructive feedback from Steven Knopoff (University of Adelaide, Australia) and four anonymous reviewers on the CSCL '95 program committee.

References

[1] Ahern, R. J. Creating creditable IT curriculum. *Systems Integration Business,* 25, June 1992, 13.

[2] Blumenfeld, P. C., Soloway, E., Marx, R. W., Krajcik, J. S., Guzdial, M., and Palincsar, A. Motivating project-based learning: sustaining the doing, supporting the learning. *Educational Psychologist*, 26 (3 &4), 1991, 369-398.

[3] Conklin, J., and Richter, C. Support for exploratory design. MCC Technical Report STP-117-85, Austin, TX, October 1985.

[1]The computer-supported collaborative learning literature has not addressed coordination issues in nearly the depth that it has communication issues.

[2]Students will not be able to see any of the outer layer.

[3]The teacher will be able to see the entire inner layer.

[4]In the only paper we could find that describes a project-based IS course, Farah [8] describes a series of projects that prepare his students for their final project, an analysis using real sponsors (our project has no required sponsors). Two other differences between the Farah projects and the UB project are worth noting: (1) the Farah projects are for a systems analysis and design course for IS majors; the UB project is for all undergraduate business students, most of whom are not IS majors, and (2) the Farah projects end with a system proposal and feasibility study (equivalent to Phase 2 of the UB project); the UB project requires students implement a prototype of the system.

[5]The project does not use real project sponsors. If more time were available for the project this would be a important extension, making the project even more realistic.

[4] Curtis, B., Krasner, H., and Iscoe, N. A field study of the software design process for large systems. *Communications of the ACM*, 31(11), November 1988, 1268-1287.

[5] Davis, L. Can education meet IS career demands? *Datamation*, 35, March 15, 1989, 65+.

[6] Doke, E. R. An industry survey of emerging prototyping methodologies. *Information and Management*, 18, 1990, 169-176.

[7] Ellis, C., and Wainer, J. A conceptual model of groupware. In *Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work*, October 22-26, 1994, Chapel Hill, NC. ACM, New York: NY, 79-88.

[8] Farah, B. N. Systems analysis and design course: a project orientation. *Education for Information*, 8, 1990, 15-22.

[9] Gentile, J. R. Grades and the purposes and practices of assessment. Chapter 13 (pp. 499-527) in *Educational psychology*, Dubuque, Iowa: Kendall/Hunt Publishing Co., 1990.

[10] Katzenbach, J. R., and Smith, D. K. *The wisdom of teams: creating the high-performance organization*. Boston: Harvard Business School Press, 1993.

[11] Koschmann, T. D. Toward a theory of computer support for collaborative learning. *The Journal of the Learning Sciences*, 3(3), 1994, 219-225.

[12] Koschmann, T. D., Myers, A. C., Feltovich, P. J., and Barrows, H. S. Using technology to assist in realizing effective learning and instruction: a principled approach to the use of computers in collaborative learning. *The Journal of the Learning Sciences*, 3(3), 1994, 227-264.

[13] LaPlante, A. Computer science gets a "reality check." *InfoWorld*, 13(20), May 1991, 50-52.

[14] LaPlante, A. Graduates say IS programs need to get down and dirty (Part 1). *Computerworld*, 25 (13), April 1991, 96.

[15] Laribee, J. F. Building a stronger IRM curriculum: views from IS managers and educators. *Information Systems Management*, 9, Spring 1992, 22-28.

[16] O'Malley, C. Designing computer systems to support peer learning. *European Journal of Psychology of Education*, 7(4), 1992, 339-352.

[17] Pea, R. D. Seeing what we build together: distributed multimedia learning environments for transformative communications. *Journal of the Learning Sciences*, 3(3), 1994, 285-299.

[18] Rein, G. L. Grades that motivate. *Proceedings of the International Association for Computer Information Systems*, Toronto, Ontario, Sept. 1995.

[19] Rein, G. L. High performance teams: a case study of a student team. Working Paper 774, SUNY (Buffalo), September 1994.

[20] Rittel, H. W. J., and Webber, M. M. Dilemmas in a general theory of planning. *Policy Sciences*, 4, 1973, 155-169.

[21] Scardamalia, M., and Bereiter, C. Computer support for knowledge-building communities. *Journal of the Learning Sciences,* 3(3), 1994, 265-283.

[22] Yaffe, J. MIS education: a 20th century disaster. *Journal of Systems Management,* 40, April 1989, 10-13.

[23] Zmud, R. W., Anthony, W. P., and Stair, R. M. The use of mental imagery to facilitate information identification in requirements analysis. *Journal of Information Systems*, 9 (4), Spring 1993, 175-191.

Author's Address

Gail L. Rein: Department of Management Science and Systems, School of Management, 310-C Jacobs Management Center, State University of New York, Buffalo, NY, 14260-4000, gailrein@acsu.buffalo.edu.