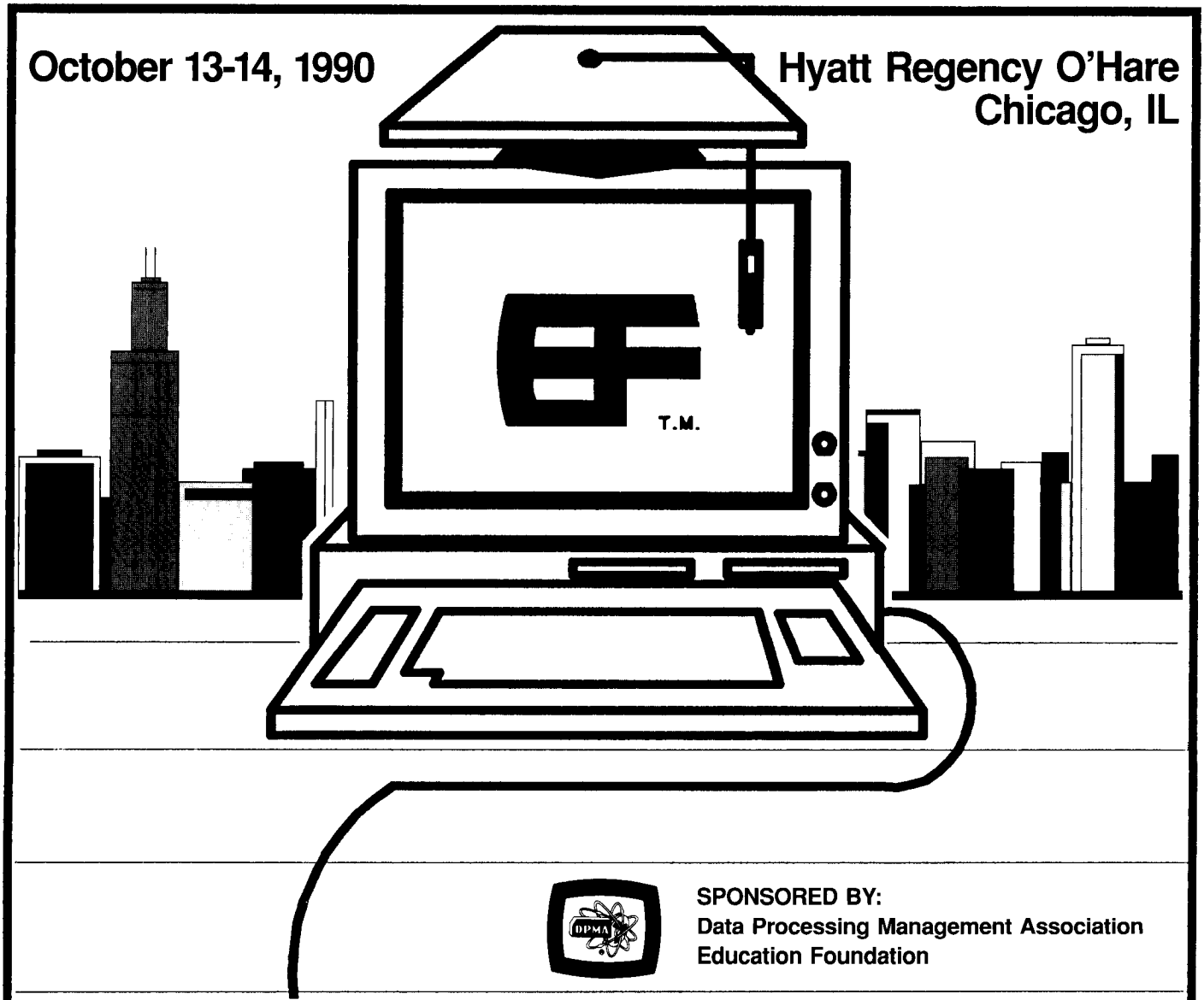
ISECON '90

INFORMATION SYSTEMS EDUCATION CONFERENCE

October 13-14, 1990

Hyatt Regency O'Hare
Chicago, IL



SPONSORED BY:
Data Processing Management Association
Education Foundation

PROCEEDINGS OF THE EIGHTH INFORMATION SYSTEMS EDUCATION CONFERENCE

"Connectivity - Plugging into IS Education"

ISSUES IN INFORMATION SYSTEMS EDUCATION

1990

The Proceedings of the Eighth

Information Systems Education Conference

October 12-14, 1990 **Chicago, Illinois**

Editor
Robert F. Zant
University of Alabama
in Huntsville

Sponsored by:

Data Processing Management Association

Education Foundation

ACKNOWLEDGEMENTS

Corporate Support

A significant grant from International Business Machines Corporation (IBM) has enabled the DPMA Education Foundation to offer reasonable registration fees for ISECON '90.

IBM has supported and followed with great interest the ISECON conferences over the past several years.

A total of 66 papers were submitted to ISECON '90. Of those submitted, 41 were accepted. Each paper was reviewed by at least two referees who submitted individual paper reviews, detailed comments, and ranking of the papers reviewed.

WELCOME TO ISECON '90

Plug into Information Systems Education by fully participating in the varied opportunities for professional growth offered by ISECON '90. This is your conference. It is designed to assist you, the information systems educators and industry trainers, in meeting the increasing challenge of guiding others.

Connect with fellow educators at both the Friday evening "Meet Your Colleagues" reception and at the Saturday Roundtable Luncheon. These events provide an opportunity to explore mutual interests with friends, both old and new.

Plug into new ideas by attending the Saturday and Sunday Paper Sessions. The Sunday Forum Session will offer you the opportunity to listen to presentations on a topic of interest and participate in the subsequent discussion. And be certain to avail yourself of the opportunity to connect with the exhibitors to obtain firsthand information on their latest offerings.

Your professional responsibilities are vast in this ever-changing information systems environment. It is the objective of ISECON to create an experience in which you may gain new ideas, new techniques, new resources and new friends to assist you in meeting the challenge.

Welcome to ISECON '90. Enjoy the experience!

Sonya 'Sam' Anderson

Sonya 'Sam' Anderson, Ph.D.
President, Board of Regents
DPMA Education Foundation

ISECON '90

STEERING COMMITTEE

Conference Chairperson

Ronald J. Kizior
Management Science Department
Loyola University of Chicago

Program Chairperson

Robert F. Zant
MIS/MSC Department
University of Alabama in Huntsville

Local Arrangements Chairperson

Leona Roen
Department of Data Processing
Oakton Community College

Placement Chairperson

William Mitchell
Computer Science
Louisiana State University - Shreveport

Exhibits Chairperson

Denise Nitterhouse
School of Accountancy
DePaul University

Marketing Chairperson

Stan Piercefield
User and Administrative Services
Public Service Indiana

REVIEWERS FOR ISECON '90

Robert Aden
Middle Tennessee State University

Susan Athey
Colorado State University

Greg Baur
Western Kentucky University

Barbara Beccue
Illinois State University

Charles Butler
Colorado State University

Carol Clark
Middle Tennessee State University

Jon Clark
Colorado State University

Eli Cohen
Bradley University

Dorothy Dologite
Baruch College - CUNY

Rick Edgeman
Colorado State University

Steve Floyd
University of Alabama in Huntsville

Donnie Ford
University of Alabama in Huntsville

Wayne Gober
Middle Tennessee State University

R. Wayne Headrick
New Mexico State University

Cary Hughes
Middle Tennessee State University

Robert Jarman
University of Alabama in Huntsville

Medhi Khosrowpour
Pennsylvania State University, Harrisburg

Gene L. Lewis
Colorado State University

Thom Luce
Ohio University

Nilakantan Nagarajan
Central Connecticut State University

Jim Nelson
New Mexico State University

Marilyn K. Pelosi
Western New England College

Joan Pierson
James Madison University

Darleen Pigford
Western Kentucky University

Sandra Poindexter
Northern Michigan University

R. A. Rademacher
Colorado State University

Herb Rebun
University of Houston - Downtown

W. S. Remington
Middle Tennessee State University

Steve Richards
University of Alabama in Huntsville

Robert Roberts
New Mexico State University

David Russell
Western New England College

Hossein Saiedian
University of Nebraska at Omaha

Shashi Shah
Hofstra University

Fan Tseng
University of Alabama in Huntsville

Joseph Williams
Colorado State University

TABLE OF CONTENTS

RESEARCH IN TEACHING IS

COMPUTING FOR THE CHANGING WORK-PLACE

The System Design Course: An Experiential Approach

John C. Shepherd
Thomas A. Pollack
Duquesne University

ABSTRACT

The System Design/Project Course should motivate MIS majors to look with anticipation toward their budding careers without jeopardizing the purposes of the course. By combining CASE tools, a fourth generation language, project-management skills, and an actual real-life project, a realistic learning environment that mirrors an actual information systems development environment can be simulated. Our paper will describe how our university has developed a system design course that integrates project management skills, CASE tools, 4GLs, database design and processing, and prototyping: Students are required to use the tools of tomorrow to design a system for today.

INTRODUCTION

A recent DATAMATION article by W. Foss preaches that "techniques that focus on generating excitement through early results in applications development can expand the potential of modern software tools." Duquesne University buys into that philosophy by offering a system design course that is thoroughly hands-on with exciting "early wins" and which uses an integrative approach where students develop an actual information system by incorporating the following skills:

- o Fourth Generation Languages
- o Project management
- o Database design
- o Database processing
- o Report design
- o Prototyping
- o System implementation using the 4GL
- o Screen/form/panel design
- o CASE tools

COURSE DESCRIPTION

Our course, which is in effect a senior-level capstone course, is entitled "MIS in Organizations". It is described in our catalog this way. "Establishes the role of management information systems and decision support systems in organizations. The impact of the information system on organizational objectives is examined. The importance of accurately defining information requirements for all levels of management in a manner that fully utilizes the capacity of the information resource is stressed."

DESCRIPTION OF THE COURSE

On the first day of class, a project is distributed to each class member and a short discussion is conducted. Projects are based on the instructor's experience or upon a system with which students should be familiar. We attempt to vary the project each semester to eliminate students using solutions from prior semesters. The project's complexity is a function of the class size, but typically consists of three subsystems, suggesting three project teams. A project manager and three team leaders are appointed or volunteer.

Next, we explain how the course will proceed by giving an overview of the course topics that we will cover or review: database design; SQL; report/form design; screen design; program design; documentation; prototyping; CASE tools; structure design tools; and a fourth generation language (4GL).

As each is discussed, we indicate that instead of just lecturing on the topics, as is done in most nonprogramming-oriented MIS courses, the class will apply the lecture material to implement the project. Each week the portion of the project relative to the previous week's material is due.

COURSE FEATURES

We have had widespread corporate acceptance of our course because of its reliance on project/team skills and the realism it brings to academia. The next several sections discuss features of the course.

Fourth Generation Language (4GL)

We firmly believe that the major benefit of the course is the class's attempt to actually implement an information system in a team environment. Teaching design topics without applying them isn't sufficient.

We are currently using the PC version of Informix-4GL because of its procedural and non-procedural programming language, its reliance on using cursors for embedded SQL, its interactive, ANSI-compliant SQL2 capabilities (ISQL), its excellent window manager, its screen painting facility and its non-procedural/procedural report writer.

Lessons Learned: By using a 4GL, most of the project can actually be implemented. As we discuss a design topic in theory, we follow that lecture with a how-to-do-it lecture using the 4GL. Because the project is team oriented, unless you have a LAN and a multi-user version of the 4GL, or use a mainframe or mini-based version of the language, multiple microcomputers will be needed, each with a redundant copy of the 4GL package and, as time goes on, the project's data and programs. This causes program and data integrity problems because each machine might have a different version of the database and/or the programs, screens, forms, etc. Be sure that this problem is discussed and a solution found. The ideal environment is to use a multi-user LAN version of the 4GL. PC versions are usually more user friendly than their larger mainframe and mini counterparts, and students are comfortable with PCs.

CASE Tools

For the past three years we have been using the version of Excelerator, which can be acquired upon adoption of the Whitten, Bentley and Barlow text, Systems Design and Analysis Methods. The students are given assignments from Whitten and Bentley's text USING EXCELERATOR FOR SYSTEMS ANALYSIS & DESIGN. We spend a portion of the second class having each student set up a project on at least two computers, and adding themselves as users of their project. We use two computers so that students have a backup machine in case one is busy. While this consumes a lot of disk space, our classes are small and the work is done in teams of two. By having the students add themselves as users (instead of us doing it), they become familiar with using a mouse as well as the look and feel of Excelerator. It can be a formidable package and by adhering to the "Early Win" philosophy, the student's trepidation is reduced.

For the first eight weeks, the students complete almost all the exercises in the Excelerator text. No class time is spent teaching them how to use Excelerator, as the text is written such that students can learn the CASE product independently. If they do encounter a problem, the Excelerator reference materials are available in the PC lab. Students can also consult with their professors.

Lessons Learned: Our experiences with the Excelerator product are quite good. We do recommend that you use a high resolution printer or the output will not match the screen diagrams because many of the words are truncated to fit

the lower resolution printer requirements. Also, be sure that all the computers use the same level of DOS, as the product uses DOS's BACKUP program, which requires that the computer on which the RESTORE is done use the same DOS level as the one on which the backup was originally made. While many schools use the Excelerator product, most don't use it to assist with an actual project. Where the Whitten text's DFD examples are already leveled and balanced, and the data modeling done, students quickly realize how difficult these tasks are when faced with having to implement an actual design from scratch.

Project Management

The project leader is responsible for:

- Establishing program naming conventions
- Choosing column naming conventions
- Choosing table naming conventions and procedures
- Development of context and level 0 DFDs
- Design of the conceptual database
- Printing of the conceptual Entity-Relationship-Diagram (ERD)
- Maintenance of a central working-paper file
- System backup
- Helping to grade team leaders

Lessons Learned: The project leader must be available to you and the students at almost all times. Don't allow the class to select a nontraditional student (i.e., one who works 40 hours a week) because those individuals aren't around the university enough. Also, the course makes huge demands on this person's time, so a weak student, regardless of his or her organizational and interpersonal skills, shouldn't be chosen either.

Make the project leader maintain a central file in the PC lab so that students have access to up to date documentation when needed.

Database Design

We begin by first listing each subsystem on the board, and then brainstorming about the transactions required for each subsystem. This is done interactively, with the team leaders in control of the blackboard, and the entire class participating.

Next, we associate entities with each transaction. When we see a duplicate entity, we don't write it on the board. Next, we assign attributes to each entity. Because the students have been through a database course, the tables end up close to Third Normal Form (3NF) or Boyce-Codd Normal Form (BC/NF). For those that don't, we intervene and make some design "suggestions." When the design is complete, all tables conform to BC/NF. Next, the class participates in developing the conceptual database using data structure diagrams.

Each team leader then formulates an external database (subschema) design on the board. Again, all of this is done in a single class meeting with all students participating.

The project manager is then given the responsibility for generating the hardcopy graphical DSD/ERD model via Excelerator. Each team leader does the same for his/her external database design. All designs are distributed to the students and part of the next class period is spent reviewing the database designs.

Once we have class approval on the designs, each team must bring test data to the following class. Now the tables are created using Informix's ISQL and data entered by means of the Informix screen generator, a facility that produces a usable data entry screen for each table without having to resort to multiple "INSERT INTO..." SQL statements. Finally, the database design, table definitions, and test data are placed into the working paper file.

Lessons Learned: Do the conceptual design as we do it. Students benefit from being involved in the design, rather than having each team leader develop the design without class interaction. You will be surprised by the excellent ideas that class members make. By participating in each step of the design process, students build pride of ownership for the evolving system. Also, this way you can guide the class toward a manageable design, one that is more closely normalized than if they performed the process without you, and you can point out any glaring weaknesses. We don't indicate the small design deficiencies, as we feel strongly that experience is, in many cases, the best teacher: Make a mistake once, and hopefully you won't make it again.

Report Design

We first cover the concept of report design. Then, based on the reports found in the list of transactions discovered in the database design step, we make the team leader assign a student to each report.

That student first designs the report the "old fashioned" way - by using printer layout charts. Or, he or she can use a word processor, which is in keeping with the trend in industry. We force each team member to be responsible for a report.

We make a copy of each report for the entire class, and ask each member to present his/her design to the class. This forces the students to stand before a group and speak, a feat most students look forward to with trepidation. The entire class is expected to comment on the design as it is presented. They usually spot inconsistencies across the reports (the same abbreviations aren't used, the run date is in a different location, the company's name is different, etc.). Once the reports are discussed, the responsible student then makes the corrections and prototypes the report using Excelerator.

The hand-drawn design and the CASE version are placed into the centrally located working-paper file in the PC lab.

Now we lecture on the report-writer for the chosen 4GL. This takes about three hours and includes many handouts of actual programs. Our text book, Building Applications Using a 4GL, by Mark Sobell, includes several good chapters on how to develop reports using Informix's 4GL, but more importantly also discusses report design in general terms.

Each student who was assigned a report then has one week to produce the initial reports. These are reviewed by the entire class and if approved, placed into the working-paper file. Now we are ready to teach screen design.

By this time, about one-third of the semester is over, but students have a good leg up on the project. They have designed the database, inserted data into all tables, and produced output. In short, by this time, they have many "early wins" under their belts.

Lessons Learned: Having the entire class participate in the report design process is invaluable, as everyone benefits from the interaction. If disputes arise while a student presents his or her design, let the class settle them, unless it appears that the discussion is circuitous or you are pressed for time. Such interactions are what the course is really all about - experiencing first-hand how a project team actually functions.

As we do, tie all the report tools and techniques together by having the class ultimately produce reports using the 4GL. That is, have them design the reports using hard copy, then prototype the report using the CASE product, and, finally, use the 4GL to produce the report from the database created during the database design phase.

Nothing excites students more than seeing quick results from topics learned in class. The advantage of using a 4GL is that reports can be quickly and easily produced. Create the database, use a default data entry screen to enter the data, then use the report writer to produce the reports.

Screen Design

Teaching screen design follows the same steps as report writing. Namely:

- Spend one lecture on how to design screens
- Spend some time in class to determine which screens are needed
- Use a structured tool, such as a dialogue chart, to document the sequence of screens
- Assign someone from each team the responsibility for designing a hardcopy version of each screen (out of class)
- The responsible student discusses the design in class
- The class makes recommendations changes
- Prototype the screens using Excelerator (out of class)
- Spend 1-2 lectures on the screen painter/window manager of the 4GL
- Students use the 4GL screen painter to actually develop the data entry screens (out of class)
- Students hand in 4GL listings plus screen dumps of each screen
- All documentation is placed in the working-paper file

As we lecture on screen design, we present handouts showing one of industry's standards for screen design, namely IBM's Common User Access (CUA), one of the components of their Systems Application Architecture (SAA) concept. Because our 4GL isn't graphics oriented, we don't expect the students to implement the elaborate graphical CUA, but we do expect them to closely follow the text-version of the CUA, whenever practical (time constraints sometimes prohibit such precise screen spacing and formatting).

Lessons Learned: Students like this part of the course because they can be quite creative. Be sure to stress that too many colors can be distracting. Also, you will need to stress the need for consistency across all student designs. Are captions to the left or on top of the screen field? How are protected/data entry fields indicated? Where will help and error messages appear?, etc.

Now the students have data entry screens and reports. All that is left is the program design step.

PROGRAM DESIGN

Our students have already had two COBOL courses, so we don't expend much effort reteaching algorithms. Instead, we focus on software engineering. For example, we have the students produce structure charts for each program. A walk-through is held and the class participates en-mass. Also, we require that for each program, at least one function must be coded using structured English.

When the modules have been fully defined, the actual programming is ready to begin. Allow about six weeks for this phase.

Through sample programs in the text and handouts from the instructors, the class is taught the 4GL programming language. Remember that the reports and input screens are already finished, so all that remains is for the students to implement the table updating routines, and to tie the disparate parts together through menus.

From this point on, we don't formally meet. Rather, we use the class time for the teams to meet to hash out problems, for the instructors to help with bugs, and to get feedback on student performance from each team leader.

Additionally, each week, the team

leaders must prepare a formal progress report, complete with a GANTT chart. Team leaders present the report to one of the instructors at a mutually agreeable time, but outside of the class room. That way, any information about laggards or other problems related to specific student performance or behavior is kept private.

Lessons Learned: Don't expect students to complete the programming effort. After all it's a project-oriented course, intended to simulate a real systems development effort. The course objective isn't to implement a system, it's to experience project-oriented work efforts, to experience the frustration of working with others, the constant changing of design specs as projects evolve, and tying together the myriad of tools and techniques taught in the other IS courses.

The importance of having someone maintain consistency across the different CPUs is essential. It isn't unusual to have students complain that their work from yesterday was wiped out due to the carelessness of someone else.

Also, expect student frustration and fear to increase as the end of the semester draws near. Don't relent: Tell them their grade depends on getting the work out. Students must be taught the importance of meeting established deadlines, and if they know in advance that you don't really expect them to finish, they won't respond to the project leader's efforts to improve their productivity.

About 10 days prior to the conclusion of the semester, we inform the class that it is time to produce the final documentation: user manuals. We provide a specific format and the final documentation is the last component of the student's grade.

CONCLUSIONS

The system design course is usually our final look at our graduating MIS students. We need to demonstrate how all the other courses they have taken fit together. If the purpose of MIS is to produce systems, then it seems to us that at some point in their career as a student, they must experience this concept first-hand. Our approach bolsters student confidence through the "early win" philosophy, provides an excellent focus for student discussions in interviews with prospective employers, enables students to participate in an actual group project, complete with all the problems associated with group dynamics and demonstrates the difficulty and importance of working with a demanding time schedule.

Such a course is never complete, and each semester we adapt it to remedy as many problems as we can. The course is quite time consuming for both faculty and student, but the rewards for both parties seem to diminish the memories of the problems, and each semester the faculty and students seem to look forward to the design course.

REFERENCES

1. Foss, W. Burry, "Early Wins are Key To Systems to Success", DATAMATION, Vol 36, No. 2, pp. 79-82.

2. Jenkins, A.M., "Prototyping: A Methodology for the Design of Application Systems," Indiana University, Bloomington IN, 1983.

3. Pendegraft, N., and Reyes, M., "A Joint Approach to Teaching User Participation in IS Development", Interface, Volume 11, Issue 3, Fall, 1989, pp.33-35.

4. Sobell, M. G., BUILDING APPLICATIONS USING a 4GL, with Examples from INFORMIX-4GL, Sobell Associates, Menlo Park, CA, 1986.

5. Whitten, J. and Bentley, L. USING EXCELERATOR FOR SYSTEMS ANALYSIS & DESIGN, 1987 Irwin, Homewood IL.

INFORMATION CONCERNING GROUP PROCESSES IS BENEFICIAL TO STUDENTS IN A SYSTEMS DEVELOPMENT COURSE

ANDREW TELLEP
THE PENNSYLVANIA STATE UNIVERSITY - SCHUYLKILL CAMPUS

ABSTRACT

Systems development is usually a group activity. For that reason, many situations that arise during the development of a system could be handled better by personnel who are aware of strategies and techniques that can make group activity more effective. Instructors in system development courses can do their students a great service by including some information concerning group processes. This information can be found in the research of educational and social psychology. The author includes such information in a systems development course he presently teaches. The result is a great reduction in the amount of time spent dealing with group problems later in the course. If students are aware of how a group functions and the problems that can arise, they are able to anticipate and compensate for those problems before they become major problems. This ability has literally saved many projects that otherwise would never have been completed. This knowledge has great value when transferred to situations after graduation, since most on-the-job projects are group efforts. The author recommends the inclusion of same basic information concerning group processes at the beginning of a systems development course. Equally important is the inclusion of some information and examples dealing with inter-group relationships and processes. This is true since few system projects are completed without the project group having to deal with other groups, i.e. management, other project groups, system users, etc.

INTRODUCTION

After many years of teaching systems development and working as a consultant in systems development, the author has come to realize the usefulness of understanding group process strategies and techniques. Perhaps having a background in education is helpful, since education majors are exposed to group process theory in the hope that it will help them manage their classrooms. It seemed natural to bring group process theory into the systems development course after seeing several project teams crumble due to their lack of cohesiveness and effectiveness when group work was required.

There are some problems with adding material to any course since the time allotted to the course is fixed. What should be added and what should be removed? The author felt that time could be saved during the course if group problems could be reduced. Problems such as a group member who does not contribute or the resolution of conflicting ideas usually find their way into meetings with the instructor. If some of these problems could be eliminated or minimized, the instructor could concentrate the time saved on teaching the traditional systems development material. Time should then be available for inclusion of new material such as group process strategies and techniques. What is unusual is that the new information is included at the beginning of the course and the time for it is made up later through a reduction in group problems that would have needed instructor attention. So, what is eventually

removed from the course are group problems requiring instructor attention.

WHAT TO INCLUDE ABOUT GROUPS

As for what should be included, the author can only suggest some topics that seem worthwhile to himself and others. They include both intra-group and inter-group process topics. In a recent article by Van Meer and Sigwart (9), a list of useful topics was discussed. They included individual performance, conflict resolution, interpersonal communication and group cohesion. These are all intra-group topics. An example of some material dealing with individual performance is provided in the following paragraph.

Individual performance can be thwarted or stimulated when an individual becomes part of a group. It is usually thwarted when the individual and the other groups members are unaware of each others needs while working together. Such needs might include acceptance of other points of view, tolerance of lesser abilities, freedom of expression, etc. Individual performance can be enhanced if group members are aware of each others needs. First, however, each member needs to be made aware that the other members have needs. An instructor can make them aware in a short period of time by using some simple examples of past experiences in group work.

Individual performance, conflict resolution, interpersonal communication and group cohesion have all been studied in depth but very little of this research is usually explored by someone educated to be a computer scientist or systems analyst. Some studies and texts dealing with group processes should be mentioned here. Brilhart (2) discusses interpersonal communication. Forsyth (5) deals with many aspects of group activity including individual performance. Fiedler and Meuwese (4) studied the contributions made by different types of leaders to group cohesiveness. There is a wide range of studies presented in Steiner and Fishbein (7) that deal with social psychology. Much of this material is devoted to the study of groups. These texts and studies are but a few of the many sources of information an instructor could use when preparing to discuss group processes in a systems development course. It is not the aim of this paper to discuss group theory and all that it entails. The aim is to suggest that the incorporation of some group process material in a system development course can be beneficial. The author also wishes to make it clear that group processes have been thoroughly researched and the research is easy to find if one ventures into the realm of educational or social psychology (1), (3), (6), (8).

A WORKING EXAMPLE

What follows is a description of what the author presents concerning group processes at the beginning of a systems development course. These are topics deemed useful by the author based on experiences in and out of the classroom. Another instructor might select slightly different material to accomplish the objective of making his/her students aware of group processes and how to effect those processes positively.

Forming a Project Group

In a systems development course, project groups can be formed in two ways. The students form their own groups or the instructor places students in groups. If students are to form their own groups, they should be told to form groups based on several factors. These include individual ability, potential for cooperation, members willingness to perform tasks assigned by consensus of opinion and leadership potential. If the instructor forms the project groups, he/she should keep the above mentioned factors in mind while doing so.

There are always class members who may be seen as not having much potential to

contribute to whatever group they become a member of. However, thoughtful selection of such a person can prove useful to some groups. This person may be willing to handle tasks that no other member wants to handle in order to be seen as useful and yet not in a position of "stress" like group leader.

Organizing Group Work

The students in a project group should be made aware of the need for group objectives and the need to organize those objectives. Group cohesiveness can be enhanced if all group members participate in the formulating of objectives for the entire group. A sense of belonging and responsibility will be developed in each member. No member will feel as if someone is dictating the group's activities.

The assignment of tasks to group members is best done in a positive sense. Students should be told to try their best to assign tasks to members based on ability, willingness of the member to accept a task, equal work assignments and a consensus of members' opinions. If there are tasks that no member wants to perform they should be shared in some democratic fashion. After tasks are assigned, a written report concerning assignments should be presented to the instructor. This helps motivate the members who might otherwise shirk their responsibilities. It also makes evaluation of individual performance easier.

Groups should be told not to select a leader until it seems necessary to the group. This may seem unusual at first but students should also be told that many groups function best under default leadership or consensus of opinion. The required selection of a group leader or the selection of a leader by the instructor can sometimes divide the group and/or alienate the leader. When a leader/organizer is needed the group will sense the need and select a leader.

Conflicts

The two concepts that can best help groups to resolve conflicts are: treat conflict as a positive entity that will lead to a needed solution and try to be considerate of all points of view. Many students believe an argument or conflict is always the same as a fight. They need to be told that argument can lead to the best solutions for all concerned. Students are usually able to understand the concept of being considerate of others' ideas but few of them know how to select the idea that is best for the group. They need to know that this is best accomplished through thoughtful, reasonable argumentation.

Keeping on Task and Walkthroughs

In group work situations it is extremely important to keep all members focused on their assignments and to have members complete their assignments on time. This is true because one member's work is usually linked to the work of other members in the project plan and must be done correctly and in a timely fashion for the project to proceed as planned. Group pressure and cooperation are both helpful in keeping members on task. Group members must expect each other to finish tasks. They must show confidence in each other and a willingness to help each other. It should be understood that help is available for all members from other members. However, this help should be volunteered and not asked for. To do this, group members must be aware of what tasks are assigned to each member and what progress has been made toward completing those tasks.

Given the information in the preceding paragraph, a student becomes more likely to understand the purpose of walkthroughs. Walkthroughs should not be intimidating but rather an opportunity to show progress and/or difficulties in order to garner praise and/or help as needed.

Inter-group Activities and Techniques

Most inter-group activities occur at the beginning and at the end of a project. Students need to know that their projects will not be done in a vacuum especially if they are involved in projects on the job after graduation. It is very beneficial for students to have to deal with other groups in order to complete their systems development projects. Therefore, if the instructor can arrange "real" projects that are of relevance to some organization or individual, it would be wise to do so. Students sometimes find excellent projects to work on by themselves, if they are encouraged and allowed to do so. These projects are much more meaningful and seem to be relevant from the students' point of view. This help enhance student interest and motivation.

If projects are done in coordination with out-of-class groups, students need to know something about inter-group processes. Such things as "office" politics, dealing with groups that are not technically oriented, and discovering key personnel in an existing system without ignoring that system's managerial structure. Knowledge of these processes is necessary during the analysis of any "real" system. They can also prove useful when a project is complete and ready for presentation. Handling the final presentation of a group's work can be crucial to the acceptance and/or rejection of that work. The group must know who to invite, how to address their audience and who they must impress in order to have their work approved. Practice with these types of group activities is very beneficial for students after graduation and discussion about having done these things can impress recruiters at job hunting time.

Communications

Students must be convinced by their instructors that communications is the key element in systems development when done as a group activity. Their systems development instructor should tell them so. If they are not told, they will learn this fact eventually, but in a slow, painful, trial and error way. Group activity and effectiveness hinges on good communications. From interviews during analysis, to written requirements, to documentation of system design, to final presentation, all success and effectiveness is reliant on good communications.

SUMMARY

If students are not made aware of some of the pitfalls of group work prior to the assignment of a group project, much time is often wasted handling those pitfalls during a systems development course. The time used to present some group process material at the beginning of a systems development course will be made up during the remainder of the course since fewer group problems will surface due to student awareness of how to make group activities effective. The location of basic group process material for use in systems courses is an easy task if one looks at the research dealing with educational or social psychology. The author teaches and does systems development and has included an example of the type of group process material he feels should be in a systems development course along with the justification for it.

REFERENCES

1. Benne, K., and Sheats, P. (1948). "Functional Roles of Group Members". Journal of Social Issues 4, pp. 41-49.

2. Brilhart, J. K., (1986). Effective Group Discussion, 5th ed., William C. Brown. Dubuque, Iowa.

3. Cartwright, D. and Zander, A. (1969). Group Dynamics. New York: Harper and Row.

4. Fiedler, F. E. and Meuwese, W.A.T., (1963). "Leader's Contribution to Task Performance in Cohesive and Uncohesive Groups". In Steiner (1965) Current Studies in Social Psychology. Holt, Rinehart and Winston. New York.

5. Forsyth, D.R., (1983). An Introduction to Group Dynamics. Brooks-Cole. Monterey, California.

6. Raven, B. (1959). Bibliography of Publications Relative to the Small Group. Technical Report, No. 1. Office of Naval Research.

7. Steiner, I.D. and Fishbein M., (1965). Current Studies in Social Psychology. Holt, Rinehart and Winston. New York.

8. Thibaut, J. (1959). The Social Psychology of Groups. New York: Wiley.

9. Van Meer and Sigwart, (1989). "Effective Group Interaction - Some Aspects of Group Projects in Computer Science Courses". SIGCSE Bulletin, (December, 1989) ACM Publications. New York.

Tools for Understanding and Representing Information Systems in an MIS Course

James A. O'Brien
Craig A. VanLengen
Northern Arizona University

ABSTRACT

Two studies were conducted in the management information systems (MIS) course at Northern Arizona University comparing the ability of students to prepare a process model (data flow diagram) and a status model (system component matrix). The first study found a significant difference in favor of the status model while the second study showed no significant difference between the two models.

INTRODUCTION

Data flow diagrams (DFDs) are the most popular graphical tool presented in textbooks for systems analysis and design courses required for CIS/MIS majors [Kievit, 1989]. Textbooks in management information systems (MIS) designed for all business administration majors present data flow diagramming as a means of documenting and increasing end-user understanding of information systems [Awad, 1988, Burch & Grudnitski, 1989, Kroenke, 1989, Laudon & Laudon, 1988, Parker, 1989]. Kuehn and Fleck [1990] also recommend DFDs as good communication tools between analysts and end users and as problem solving tools. However, since the majority of students in the required business core MIS course using these textbooks are non-CIS/MIS majors, it is possible that tools other than data flow diagramming may be more appropriate for such students. This position is based on the authors' belief that a proper understanding and ability to use data flow diagramming may require the students to document several systems over several weeks. This is about the amount of time and effort spent on data flow diagramming in a systems analysis and design course for CIS/MIS majors.

Given the more business end user and managerial orientation of the MIS course and its student audience, the authors question the appropriateness of allocating this amount of time to learning data flow diagramming or any other systems development tool. O'Brien and VanLengen (1988) proposed the use of "status models" for non-CIS/MIS majors to aid students in identifying the status of components of an information system. The status model proposed is a system component matrix, also found in O'Brien (1990). Exhibit 1 shows that the system component matrix is a status model that shows the resources, products, and activities of an information system at a specific point in time. A data flow diagram, on the other hand, is a process model, ie., it shows how data flows between the processes and resources to produce the end products. The authors conducted a study to ascertain if students in a junior level MIS course were able to understand a system better using a status model or a process model.

RESEARCH APPROACH

During the Spring semester 1989 all students in two sections of the CIS 360 Management Information Systems course at Northern Arizona University were given the same amount of instruction in preparing a process model (data flow diagram) and a status model (system component matrix). After this instruction students were given the same

narrative description of an actual computer-based information system of a business firm. The students were randomly assigned to complete either a process model or a status model from this narrative. Student demographic data is presented in Table 1.

Both groups were given reference material on the symbols and general format for the model they were assigned. The models produced by the students were assigned a score based on the percentage they completed compared to models specified in an answer key prepared by the instructors. An analysis of variance was used to test the following research hypothesis:

H_0: There will be no difference between the completion scores of the students completing the process or the status model.

RESULTS

The mean and standard deviations are presented in Table 2. The analysis of variance for the percent score on model construction is presented in Table 3. The results show a significant difference in favor of the status model group. This allows us to reject the null hypothesis.

DISCUSSION

The results indicate that the status model was easier to complete than the process model for students taking the CIS 360 Management Information Systems course at Northern Arizona University. A critique of the results was then conducted to determine if any biases or errors were made during the experiment in favor of the status model. Instruction and wording of the system narrative were determined to be areas where bias could have been entered into the experiment. The experiment was repeated in the Fall of 1989. Again all students in two sections of CIS 360 Management Information Systems course were given the same amount of instruction in preparing a process and a status model. Student demographics are presented in Table 4.

This time the process model instruction was presented by an instructor who teaches a systems analysis and design class for CIS/MIS majors. The problem statement was revised for the requirements that were found to be more difficult for the process model group in the previous experiment.

RESULTS OF THE SECOND STUDY

The means and standard deviations are presented in Table 5. The analysis of variance for percent completion on model construction is presented in Table 6. The results of the second experiment failed to reject the initial hypothesis. The status model group performed better than the process model group but not at a significant level.

DISCUSSION OF BOTH STUDIES

The results of the two experiments give inconclusive findings on the advantages of the status and the process model representations of information systems for students enrolled in CIS 360 Management Information Students at Northern Arizona University. Therefore the study results will be further analyzed to ascertain if the narrative that was rewritten for the second study became more biased in favor of the process model group. One example from the narrative used in the first study was a description of the use of data communications to transmit backorders to an entity outside the scope of the system being documented. A large number of the students in the first study represented this as a process of the system and not as an external entity. Other narrative statements will be evaluated to either locate or develop a system description that is neutral between the two models. Instruction in both models also needs to be evaluated to ensure that it is equivalent. Future studies should also look at the ability

of students in the MIS course to construct and then to use the different models to identify benefits, limitations, and business problems or opportunities provided by an information system.

CONCLUSION

Since more Business Administration programs are making the MIS course an upper-division requirement for all Business Administration majors, new tools must be found to help non-CIS/MIS majors understand and represent information systems in the real world. The authors propose the use of status models, such as the information system matrix, instead of process models like the data flow diagram. Preliminary findings of research studies indicate that non-CIS/MIS majors may find tools like the system component matrix easier to use than more traditional systems development tools such as the data flow diagram. Future research is needed to clarify these tentative findings and explore the effects of both types of tools on the ability of non-CIS/MIS majors to identify and propose information systems solutions to business and management problems in the real world.

REFERENCES

Awad, E. M. (1988). Management Information Systems: Concepts, Structure, and Applications. (pp. 432-436). Menlo Park: Benjamin Cummings.

Burch, J., & Grudnitski, G. (1989). Information Systems: Theory and Practice. (pp. 234-239). New York: Wiley.

Kievit, K. (1989). Are systems flowcharts really going, going, gone?, Interface: The Computer Education Quarterly, 11(3), 17-22.

Kroenke, D. (1989). Management Information Systems. (pp. 411-426). Santa Cruz: Mitchell.

Kuehn, R. R., & Fleck, R. A., Jr. (1990). Data flow diagrams for managerial problem analysis. Information Executive, 3(1), 11-15.

Laudon, K. C., & Laudon, J. P. (1988). Management Information Systems: A contemporary perspective. (pp. 351-358). New York: Macmillan.

O'Brien, J. A., & VanLengen, C. A. (1988). Using an information system status model for systems analysis and design: A missing dimension. CIS Educator Forum, 1(2), 21-27.

O'Brien, J. A. (1990). Management Information Systems: A managerial end user perspective. (pp. 111-113). Homewood: Irwin.

Parker, C. S. (1989). Management Information Systems: Strategy and Action. (pp. 109-111). New York: McGraw-Hill.

TABLE 1. Student Demographics Spring 1989

Group	Sex		Major		Grade Level			
	M	F	CIS/MIS	Non-CIS/MIS	So	Ju	Sr	Grad
Status (n=34)	21	13	7	27	1	9	21	3
Process (n=32)	17	15	3	29	0	12	18	2
Totals (n=66)	38	28	10	56	1	21	39	5

TABLE 2. Percent Score on Model Construction Spring 1989

Group	Mean	Standard Deviation
Status (n=34)	41.95	8.75
Process (n=32)	28.86	8.70
Total (n=66)	35.60	10.88

TABLE 3. Analysis of Variance Spring 1989

Source of Variation	Sum of Squares	df	Mean Square	F	Sig
Treatment	2,823.65	1	2,823.65	37.097	.000
Residual	4,871.42	64	76.12		
Total	7,695.07	65	118.39		

TABLE 4. Student Demographics Fall 1989

Group	Sex		Major		Grade Level			
	M	F	CIS/MIS	Non-CIS/MIS	So	Ju	Sr	Grad
Status (n=29)	16	13	2	27	0	3	22	4
Process (n=24)	18	6	1	23	0	6	15	3
Totals (n=53)	34	19	3	50	0	9	37	7

TABLE 5. Percent Score on Model Construction Fall 1989

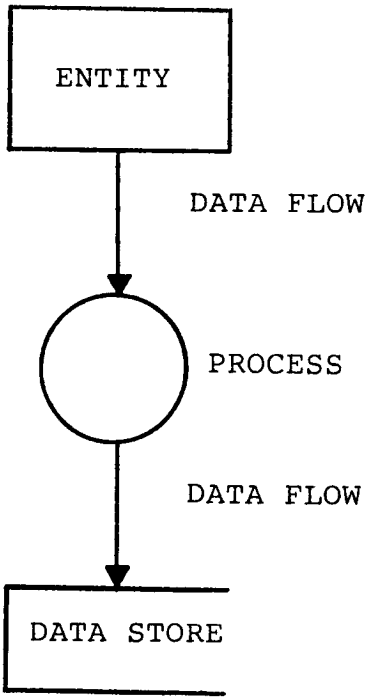
Group	Mean	Standard Deviation
Status (n=29)	43.92	14.05
Process (n=24)	42.42	11.48
Total (n=53)	43.24	12.85

TABLE 6. Analysis of Variance Fall 1989

Source of Variation	Sum of Squares	df	Mean Square	F	Sig
Treatment	29.406	1	29.406	.175	.677
Residual	8,555.701	51	167.759		
Total	8,585.107	52	165.098		

EXHIBIT 1

PROCESS MODEL
DATA FLOW DIAGRAM



STATUS MODEL
SYSTEM COMPONENT MATRIX

Information System Activities	Hardware Resources		Software Resources		People Resources		Data Resources	Information Products
	Machines	Media	Programs	Procedures	Specialists	Users		
Input								
Processing								
Output								
Storage								
Control								

A COMPARISON OF INTERACTIVE PROGRAMMING METHODOLOGIES BETWEEN THE C AND BASIC PROGRAMMING LANGUAGES

Gerald J. Tatar
School of Business
Duquesne University

ABSTRACT

The purpose of this research was to structurally compare the BASIC and C programming languages for computer based instruction applications in the qualitative areas of interactivity level, screen design, self pacing, record keeping, random selection of test questions, and tutorial level and in the quantitative areas of the number of GOTO's used, program time development, the number of lines of coding used to write a CBI program, and the number of lines of coding used for data validation. A tutorial program was constructed in each language by each group. The groups consisted of a class of non-computer science majors at one north eastern university and a class of computer science majors at a second north eastern school. The main findings were: The programs written in C used statistically fewer GOTO's than their BASIC equivalents. The interactivity level, self pacing capability, and record keeping capability proved to be statistically better in the C programs than their BASIC counterparts. There was no statistically significant difference between the languages in the other areas investigated.

INTRODUCTION

Lack of portability of instructional programs is a problem encountered today by the instructional design technologist in the development of computer based instruction (CBI). Portability of a computer program means that the source code can be placed on any computer and compiled with no errors with little or no modification to the source code. Selection of a computer language or authoring system can be perplexing for the instructional designer with the proliferation of the microcomputer in universities, colleges, private schools, and public schools and now with the entrance of interactive video machines into the marketplace. One possible alternative to traditional computer languages and authoring systems used in the development of computerized instructional design systems is to use the C programming language.

BACKGROUND

Ken Thompson, a Bell Laboratories scientist, developed the transportable language B. The B language was modified by Dennis Ritchie, who developed the C language, and then rewrote UNIX software in C. (3) "C is a small language. C has fewer keywords than PASCAL, where they are known as reserved words, yet it is arguably the more powerful language." (1)

"C is a language that can work scientific, accounting, and data processing problems with equal effectiveness." (2) "C is portable. C is portable by virtue of being small and by initially being defined on a small machine, a PDP-11." (1) C is modular. C is not without criticism. It is not as strongly typed as other recent programming languages. It allows the compiler to reorder evaluation within expressions and parameter lists. It has no automatic array bounds checking. It

makes multiple use of such symbols as "*" and "=".

PURPOSE OF THE STUDY

This study investigated those factors relative to the potential success of the C programming language in the instructional designer's task of designing interactive programs to use with computer based training devices. It has attempted to identify various standardized objective criteria to be used in comparing the C programming language to the BASIC programming language as a viable alternative to designing interactive computer based instruction. Individuals were screened relative to their prior exposure to these languages. Entering attributes such as previous programming experience were documented.

HYPOTHESES

H:1 There is no significant difference in the mean number of "GOTO" instructions used to author equivalent CBI programs in the C and BASIC programming languages.

H:2 There is no significant difference in the mean time required to develop an interactive CBI program in the C and BASIC programming languages.

H:3 There is no significant difference in the mean number of instructions required to write equivalent programs in the C and BASIC programming languages.

H:4 There is no significant difference in the mean number of lines of data validation code required to write equivalent programs in the C and BASIC programming languages.

H:5 There is no significant difference in the mean interactivity level between the C and BASIC programming languages for authoring CBI programs.

H:6 There is no significant difference in the mean screen design of a CBI program between the C and BASIC programming languages.

H:7 There is no significant difference in the control of self-pacing for CBI programs written in the C and BASIC programming languages.

H:8 There is no significant difference in the support of student record keeping for CBI programs constructed in the C and BASIC programming languages.

H:9 There is no significant difference in the mean quality of the random selection of test questions for CBI programs written in the C and BASIC programming languages.

H:10 There is no significant difference in the mean construction of CBI tutorials written in the C and BASIC programming languages.

METHODOLOGY

This study was conducted to determine the efficacy of the C programming language in relation to computer based instruction. The C programming language was structurally compared to the BASIC programming language using various attributes as indicated in the hypotheses given above. A quasi-experimental research design was used in this study to best utilize the SPSS-X analysis program used by the writer to statistically analyze the data.

The population for this study consisted of those students enrolled in Computer Science/Business Information Systems (BIS) classes at liberal arts and business Colleges and Universities. The student was not necessarily a computer science or BIS major.

The sample included those students enrolled in a section of the Technological Literacy class at one north eastern university (U1) and a computer class at a second north eastern school (U2) during the fall 1986 term.

The combined subjects totaled forty students for this study. Student backgrounds were varied in relationship to age, major (computer science or non-computer science), and amount of previous exposure to CBI (If any exposure was claimed). No students had any previous exposure to C. No students in either group had any previous formal exposure to CBI.

There was one major dependent variable in this study. This dependent variable is a structured or unstructured CBI program as developed by the student in the BASIC and C programming languages. The independent variables in this study related to the CBI program constructed by the student. These include the number of "GOTO" instructions, the time to develop the program, the length of the program in reference to the number of source coding lines, and so on as indicated in the hypotheses. The hypotheses posed in this study suggest the experimental research design shown in figure 1 below. The treatment and groups are described in the following definition of variables.

1. Types of treatment:
   B = BASIC language CBI treatment
   C = C language CBI treatment

2. Groups of learners:
   U1 = First school student group
   U2 = Second school student group

3. Other definitions:
   S = Structured CBI program
   U = Unstructured CBI program

Figure 1

Result	Group	Treatment
U1	BASIC CBI	S or U
U1	C CBI	S or U
U2	BASIC CBI	S or U
U2	C CBI	S or U

This design was used to determine the effect of two languages on constructing structured CBI programs as measured by the constructs given earlier.

The Apple IIe microcomputer was used by the participants in this study. Students in a section of a CBI class in the U1 group and U2 group were used to carry out this study. Students were given instruction on developing a highly interactive CBI program in BASIC on an Apple IIe microcomputer. Students in both groups were then given instruction on programming in the C programming language.

A panel of experts consisting of full-time computer science professors at the U2 institution completed a rating form for each of the students in this study. This form was used to structurally compare the BASIC and C programming languages used in the CBI programs written by the students and to determine the applicability of the C programming language to CBI.
The Statistical Package for the Social Sciences (SPSS-X), available to the writer on a minicomputer system at the U2 institution, was used to determine the statistical measurements for this study.

A special form was distributed to the student to log the time required to complete the CBI project in the BASIC and C programming languages and to collect the data.

To test each of these hypotheses data was gathered on the student CBI project information form and the panel of experts information form for this study.

To analyze each of these hypotheses the t-test statistic was used with a p-value to test the sample means. For example, the following was tested for hypothesis number one. (i.e. U(# GOTO's in C) = U(# GOTO's in BASIC))?

An analysis of variance between and

within groups was done. Data obtained from this study was analyzed by calculating means to establish central tendencies, and standard deviations were calculated to construct t statistics to test the significance of group means between the above named constructs in both the BASIC and C programming languages. In addition, an analysis of variance was performed on the data to determine group variations.

REVIEW OF THE FINDINGS

The findings of this study are given below. The writer will first state the findings for the quantitative factors. Secondly, the findings for the qualitative factors will be presented.

1. Number of "GOTO" instructions. There was a statistically significant difference for the number of "GOTO" instructions used in both the U1 group and the U2 group. The null hypothesis is rejected.

2. Time to develop CBI programs. There was no significant difference in the time required to develop equivalent CBI programs in BASIC and C in both the U1 and U2 group. The null hypothesis is not rejected.

3. Number of program instructions. There was no significant difference for the number of program instructions used in the U2 group. There was a statistically significant difference for the number of program instructions used in the U1 group. The null hypothesis is accepted.

4. Number of data validation coding lines. There was no significant difference in the number of data validation coding lines required for the U2 group. There was a statistically significant difference in the number data validation coding lines required for the U1 group. The null hypothesis is rejected. The findings for the qualitative factors are presented below. Findings are stated for both the U1 group and the U2 group from the student's perspective and the expert's perspective.

5. Interactivity level. There was a statistically significant difference for the interactivity level in the U1 group, the U2 group, and from the expert's perception of both the U1 and U2 student groups. The null hypothesis is rejected.

6. Screen design. There was no significant difference for the screen design in the U1 group, the U2 group, or from the expert's perception of these two groups. The null hypothesis is accepted.

7. Self pacing. There was no significant difference for self pacing in either the U1 group or the U2 group from the student's perspective. There was a statistically significant difference for self pacing from the expert's perception in both the U1 group and the U2 group. The null hypothesis is rejected.

8. Record keeping. There was a statistically significant difference for record keeping capabilities in the U2 group from the student's perspective. There was no significant difference for record keeping capabilities in the U1 group or from the expert's perspective of these two groups. The null hypothesis is accepted.

9. Random selection of test questions. There was no significant difference for random selection of test questions in the U1 group, the U2 group, or from the expert's perception of these two groups. The null hypothesis is accepted.

10. Tutorial level. There was no significant difference for the tutorial level in the U1 group, the U2 group, or from expert's perspective of these two groups. The null hypothesis is accepted.

The writer will present the reader with

the conclusions of this study.

CONCLUSIONS

The first and most important overall conclusion emanating from this study is that in most of the quantitative and qualitative areas investigated, there is no statistically significant difference between C and BASIC for student construction of CBI programs. This study authenticates the efficacy of the C programming language for instructing students to use interactive structured programming techniques in the development of computer based instruction for the elements investigated.

If other elements are considered such as compilation or ease of use, the investigator may find different results. On those constructs for which there was no difference from any of the participants' perspectives the writer concludes that C is no better than BASIC nor is C any worse than BASIC.

OBSERVATIONS

The writer observed a number of items during the course of this study. The C compiler overworked the Apple IIe microcomputer's disk drives. The compiler will start and stop the drives several hundred times during the compilation of a lengthy program. The writer experienced the failure of an analog card in drive two of the Apple IIe microcomputer used as a result of this heavy use. The error messages were many times deceiving rather than helpful in aiding the student in program debugging tasks. An error free compilation is required before the student can run a program. With instructional programming this presents a burden to the student and creates a frustrating experience. It was also frustrating to the student to have to wait a substantial amount of time between compilations to be able to trial run their programs. Since BASIC is an interpretive language the students could trial run their programs while constructing them. This is not the case with C. Students had to write and key in their entire program first. The writer stresses these statements as a positive advantage with BASIC as a CBI language. The writer found two problems (bugs) with the C compiler used in this study. One problem was with the "float" variable declarator for single precision numerical storage. It did not work properly. To compensate for this problem the writer required all students to use "double" to declare numerical variables as double precision floating point values. The second problem occurred when the writer tried to use the random number generator in a subroutine. The random number generator would only generate a one. To compensate for this problem the writer instructed the students to place the random number generator in the main body of the program. It would then operate flawlessly. The writer constructed some thirty C programs during the course of this study on the Apple IIe microcomputer. The writer uploaded the source code of all thirty programs onto a minicomputer computer system at the U2 institution. All thirty programs compiled perfectly with the C compiler on this system with the exception of one program that used the random number generator. After a minor adjustment was made, this program also compiled with no errors. This demonstrates the portability of the C programming language between these two computer systems. An interesting quirk the writer identified during the course of this study is that the "scanf" input function in C will not permit a two word response as an answer in a CBI program. As soon as a blank space is entered the computer will end the character input. To compensate for this dilemma the writer was able to use the "gets()" function in C which does permit embedded blanks in the course of entering a value.

SUMMARY

In summary this study authenticates that C can be used in applications to CBI for instructing students on interactive programming techniques as investigated on the aforementioned constructs within the realm of considerations and findings previously mentioned. More work could be done to fully authenticate the efficacy of the C programming language to applications in computer based instruction.

REFERENCES

(1) Kelley, Al and Pohl, Ira, A Book On C: An Introduction To Programming In C, Benjamin/Cummings Publishing Co., Inc., Menlo Park, California, 1984, pp. 1-2.

(2) Swartz, Ray, Doing Business With C, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1989, p. 3.

(3) Thomas, Rebecca and Yates, Jean, A User Guide To The UNIX System, Osborn/McGraw-Hill, Berkeley, California, 1982, pp. 6-7.

COBOL ON MICROCOMPUTERS: A QUALITY CURRICULUM OPTION FOR TEACHING BUSINESS ORIENTED APPLICATION PROGRAMMING

Michael J. Payne
Mark W. Smith
Department of Computer Technology
Purdue University

ABSTRACT

PC based COBOL compilers are an up and coming alternative for software development in industry. In fact, many companies are now doing much of their COBOL program development on PCs. However, the academic world is still teaching most of their COBOL software development courses using the mainframe. Shouldn't the academic world also be looking at this new alternative for use in the classroom? We think so, because it is our job to educate people for industry. It is also our job to facilitate learning for the student. PC-based COBOL makes this possible by allowing us to educate our students in an upcoming area of applications development and by allowing us to use PCs to do it. This paper looks at PC-based COBOL compilers and explains why they are a good alternative for a three course sequence in COBOL. Three such reasons why they are a good alternative are that: 1) they allow entry level students to work in the more familiar PC environment; 2) they provide better interactive debugging tools; and 3) they provide a PC based environment to learn CICS. The paper also looks at the advantages (mentioned above) and disadvantages, such as technical incompatibilities between PC and Mainframe, associated with these compilers. Lastly it will recommend topics for each of the courses in a three course COBOL sequence.

INTRODUCTION

What's new but old? COBOL on the PC! Now the most widely used (and still used!) business programming language has started something new. For years COBOL has been THE language used in large mainframe environments. Granted it is wordy and slow to adapt to new standards; but it remains one of the most powerful (albeit slow and cumbersome) languages for programming business applications. So, what could be done to improve the use of COBOL? Enter the PC.

Over the years, the PC has steadily proven its worth in business. Stand alone applications such as spreadsheets and word processors have proven the value of PC's in business. Now that the PC has matured and become extremely powerful (both in processing and memory), business program development using COBOL seems a natural extension. Many companies are now exploring and using Micro-based COBOL development (1). One company has even saved $20,000 per month through the use of PC-based COBOL on a LAN! (7) There are many advantages and some disadvantages in using a PC-based COBOL. This paper will not only look at these, but it will also look at the reasons for incorporating PC-based COBOL in the college or university business programming curriculum.

A LOOK AT PC-BASED COBOL

In general, PC-based COBOL compilers

support most major language components used on mainframe versions. Figure 1 lists the main differences between micro- and mainframe-based COBOL. For a more in depth comparison see Dr. Harry C. Benham's article on "Utilizing COBOL in the PC Environment" (3).

Performance of PC-COBOL compiled programs compares very favorably with the same programs run on a mainframe (5). The major performance degradation appears when larger files are involved or when COBOL Sorts are used. However, larger memory configurations on PC's today can and will enhance performance.

Also, there are advantages in using PC-based COBOL and a few disadvantages. These will be discussed briefly in this section of the paper. For a summary see Figure 2.

Primarily the debugging tools, editors and screen generators are superior to mainframe COBOL compilers. They perform in an integrated environment and come with the compiler as opposed to being purchased add-ons as with most mainframe-based compilers. The newer PC-COBOL compilers are even compatible with both the COBOL 74 and COBOL 85 standards. Especially useful are the debugging tools for PC compilers. One such tool, the ANIMATOR, included with the Micro COBOL/2 Workbench allows the programmer to watch a listing of his/her program execute on the screen at several different speeds with the ability to open windows and watch the data variables change during the execution. Other PC-COBOL compilers offer similar features.

The main disadvantages of PC-based COBOL compilers deal with areas of technical compatibility between the PC and mainframe. Due to different hardware and software environments and the sophistication levels of the operating systems, tweaking must be done when porting PC developed programs to the mainframe. Data and file compatibility may not be the same. Thus, the programmer should expect to do some maintenance.

With these differences, advantages and disadvantages, why propose using PC-based COBOL throughout the business programming curriculum? Simply put, it's felt that it will make learning COBOL easier.

PC-BASED COBOL ACROSS THE CURRICULUM

PC-based COBOL fits nicely into a three course sequence of business programming classes. As we discuss each course in the sequence, we also will discuss the purpose behind using PC COBOL. The first or introductory course in COBOL should present a friendly environment in which the students can learn the basics of the language. Since most students are currently micro-computer literate, PC-based COBOL is a natural fit. Also since most of these students already know how to use a word processor and can use it to key in their programs, they don't have to waste time learning an often difficult and clumsy editor that comes with a mainframe. Finally, since the major goal of an introductory COBOL class is to teach the basics of the language, the students are free to concentrate on just that and not on being frustrated with also having to learn how to use a mainframe. PC-based COBOL compilers also follow ANSI standards and thus are just as good as the mainframe versions in this respect.

The intermediate level or second course using COBOL also could benefit from using PC-based COBOL. In this course the emphasis is normally placed on debugging and more advanced file techniques. PC-based COBOL supports both course goals, especially debugging, which is one of PC-based COBOL's major advantages. As to the more advanced file techniques, these can be taught using either the PC or the mainframe. Most likely it would be taught on the mainframe since more and more companies are using PC-based COBOL to do their development work and then uploading it

to the mainframe. What better way to train the students for the job market. It also allows for a natural transition to introduce the students to the mainframe. By uploading their programs to a mainframe, they can begin to learn the intricacies of mainframe programming, including the use of JCL and mainframe file environments along with file conversions from the micro-based file systems to mainframe access methods such as ISAM and VSAM.

In the advanced level course, advanced topics could easily be taught. PC-based COBOL could be used to develop CICS applications and IMS database applications. It also can be used with 4th generation languages. All three of these items are supported by at least one of the PC-based COBOL compilers on the market today. One of the specific advantages in using a PC-base COBOL for CICS and IMS/Database is the freeing up of mainframe resources and the required support personnel. Such problems found in CICS program development as with bringing down the entire system are eliminated since each student will be running in his/her own environment. Also, with today's new generation of CASE tools for microcomputer systems, actual program development could be easily done in an integrated environment not available on mainframes. Of course, ultimately code generators from some PC-based COBOL compilers could be used, and students could then modify the produced code. All these then lead to a student more fully trained for the job market.

CONCLUSION

As microcomputers have become more powerful, PC-based COBOL promises to be an exciting alternative to mainframe COBOL program development. Most of the features found with mainframe versions of COBOL are available on today's micro based COBOL compilers. Enhanced features from debugging, screen layout, and better editing improve the quality of student's programs. All of these features are integrated providing an efficient complete programming package. Using a three course approach to train students to program using PC-based COBOL offers distinct advantages over the traditional all mainframe approach. Better programming done more efficiently is the expected result. Since, COBOL programming will be around for yet a few (maybe longer) years, why not use more advanced tools found with PC-based COBOL to train your students.

REFERENCES

1. Angus, Jeff. "Info Center Managers Increase PC COBOL Use: Compiler Enhances Productivity," InfoWorld, Vol. 9, No. 7 (Feb 16, 1987), pg.33.

2. Benham, Harry C. "Utilizing COBOL in the PC Environment," Information Executive, Vol. 2, No.3 (Summer 1989), pp. 59-61.

3. Benham, Harry C. "PC Based COBOL Instruction," ISECON'88 Conference Proceedings, (Fall 198), pp. 55-58.

4. Jalics, Paul J. "COBOL on a PC: A New Perspective on a Language and Its Performance," Communications of the ACM, Vol. 30 No.2, (Feb. 1987), pp. 142-154.

5. Mirecki, Ted. "COBOL Performs," PC Tech Journal, Vol. 3 Nos. 6-8 (June, July, August 1985), pp. 58-75, 111,131,107-136.

6. DeWolf, Mary. "COBOL in a PC Setting," PC Tech Journal, Vol. 5 No. 1 (January 1988), pp. 130-141.

7. Zeimetz, Jordene. "Professional Viewpoint," PC Tech Journal, Vol. 7 No. 2 (February, 1989), pp. 160-161.

PC-based COBOL	Mainframe-based COBOL
Highly integrated set of tools for program development	Development tools often missing or extra cost
Enhanced screen building	Simple ACCEPT and DISPLAY
Easy device and file access	Device and file access dependent on Op. System control language
Collating done in ASCII	Collating done in EBCDIC
Arithmetic data type emulated	Arithmetic data types developed for mainframe processors

Figure 1.

Advantages	Disadvantages
Integrated Environment	Lack of Technical Debugging Tools
Speed	Coordination of Development Team
Interactive Source Level Debugging Tool	Technical Incompatibilities Between PC and Mainframe
Code Generation	
IMS and CICS support	
Linkage to Application Development Languages	

Figure 2.

COBOL-85 and Structured Programming: Opportunities to Improve Programming Style

R. Wayne Headrick and Robert S. Roberts
Business Computer Systems
New Mexico State University
Las Cruces, New Mexico 88003-0001

ABSTRACT

Since the introduction of structured programming techniques, their acceptance has increased to the point that most computer professionals take for granted the benefits they provide. The ANSI COBOL-68/74 standards defined a language with sufficient power to enable programmers to develop large, complex systems while adhering to the principles of structured programming. The introduction of the COBOL-85 standard, with its new capabilities, has provided the programmer with many new alternative solutions to programming problems that can be legitimately categorized as being structured. It is the thesis of this paper that the existence of these new alternatives create a situation in which the information systems professionals who recommend coding standards (i.e., development managers, authors of COBOL textbooks, educators, etc.) must consider how the new features should be used to best improve the code being produced. To illustrate this need for new guidelines, this paper discusses the coding alternatives provided by one of the many important new features of COBOL-85 and makes some recommendations regarding those alternatives as they relate to structured programming standards.

INTRODUCTION

In the development of business information systems, various structured programming techniques have been adopted with the goals of improving programmer productivity and reducing systems development costs by producing programs that are more reliable and easier to understand and maintain. Since the theoretical basis for structured programming techniques was presented, it has advanced far beyond simply programming using only sequential, iterative and selection control structures. The concept of top-down modularity, with modules exhibiting high levels of cohesion and low levels of coupling, has gained wide acceptance [1,2]. It is also generally accepted that modules should be relatively short (e.g., a maximum of 20 to 50 lines) [5,6]. To reduce the complexity of individual modules, and, by extension, the whole program, it has been suggested that the nesting of control structures should be minimized [3], and that logical control activities should not be placed in the same modules with data manipulation activities [4,6]. Adherence to these and other such standards has generally resulted in the development of code that is much more easily maintained than that developed previously.

Because the great majority of business information systems have been written using COBOL-68/74, professional business programmers have developed programming techniques that allow them to make use of the statements available under those versions to write structured, modular, easily understood and maintained code. Because COBOL-85 provides for the upward compatibility of previously written code, new code written using the old standards will still work under COBOL-85. However, a number of the new features included in COBOL-85 are equivalent to those that have proven to be extremely popular in other programming languages such as BASIC, Pascal, Modula2 and C. As a consequence, it is almost certain that their inclusion in COBOL-85 will almost certainly be greeted with enthusiasm.

In the business environment, it is the implicit intent of the inclusion of these new features that they will

be used by system developers to improve the structure, style, reliability and maintainability of their code and, in turn, their productivity. As these new features gain in popularity and usage, it is important that their use results in code that is at least as easily understood and maintained as that previously written. In fact, if the code constructed using the new features is not better than that which doesn't use them, there is nothing to be gained from their adoption. To ensure that this happens, information systems professionals must explore the many options available to system developers under COBOL-85, identifying those that have the capability to enhance code and developing standards that will lead to their best use.

To demonstrate how these new features might be used to produce *better* code, one of them, the in-line PERFORM, has been selected for a detailed discussion. COBOL uses the PERFORM statement to implement the conditional loop/iteration control structure, an important feature of nearly all programming languages. In languages ranging from BASIC to Modula2, we see statements like DO...UNTIL, WHILE...WEND, or LOOP and EXIT used to implement this control structure. COBOL-68/74 implemented this structure with the somewhat awkward PERFORM *procedure-name* UNTIL *condition* statement. Because of the significantly increased power and flexibility given the PERFORM statement under COBOL-85, this paper focuses on this particular statement, demonstrating the flexibility of its new features and discussing some of the implications of the new coding alternatives they make possible.

THE IN-LINE PERFORM

Under COBOL-68/74, the PERFORM...UNTIL statement is used to direct the iterative execution of a sequence of statements contained in a procedure external to the current logical flow of the program. While this same technique can still be implemented in COBOL-85, an in-line PERFORM...UNTIL, with accompanying END-PERFORM, has been added. Using this feature, the sequence of statements to be iteratively executed may be physically located within the current logical flow. An overview of the implementation of the PERFORM...UNTIL using both techniques is provided in Figure 1.

Using this and other capabilities available under COBOL-85, it is possible to write the entire program (PROCEDURE DIVISION) using in-line code. Although it can be argued that such an approach would not actually eliminate modularity within a program, it would make the modules considerably harder to locate. It would also result in redundant code in those situations where a particular task is required to be accomplished more than once in a program.

COBOL-68/74/85	COBOL-85 only
PERFORM <paragraph-name> UNTIL <condition>. <paragraph-name>. <statement-1> <statement-2> <statement-3> : <statement-n>.	PERFORM UNTIL <condition> <statement-1> <statement-2> <statement-3> : <statement-n> END-PERFORM

Figure 1. The PERFORM-UNTIL under COBOL-68/74/85

AN EXAMPLE

To illustrate the alternatives that this particular new feature of COBOL-85 provides, three examples of COBOL-85 code that accomplish the same task are provided below. The selected task is the printing of a report using data that was previously accumulated in a 15 by 10 working storage table called WS-DATA-TABLE. To accomplish this task, a WS-ROW-DETAIL report line record is defined to contain:

(1) a label identifying the particular row being printed,

(2) 10 positions in which the data from the entries in a row of the table are placed, and

(3) a position for a row total.

The working storage descriptions for these two data structures are shown in Figures 2 and 3 below.

```
01  WS-DATA-TABLE.
    05  WS-DATA-ROW              OCCURS 15 TIMES
                                  INDEXED BY ROW-INDEX.
        10  WS-ROW-LABEL             PIC X(5).
        10  WS-DATA-ENTRY            OCCURS 10 TIMES
                                      INDEXED BY COLUMN-INDEX
                                      PIC 9(4)  VALUE ZEROS.
    05  WS-ROW-TOTAL-ACCUM           PIC 9(5).
```

Figure 2. Accumulator Table

```
01  WS-ROW-DETAIL.
    05  FILLER                       PIC X(5).
    05  RD-ROW-LABEL                 PIC X(5).
    05  RD-TABLE-COLUMN          OCCURS 10 TIMES
                                  INDEXED BY COLUMN-OUT-INDEX.
        10  FILLER                   PIC X(2).
        10  RD-TABLE-ENTRY           PIC Z(3)9.
    05  FILLER                       PIC X(5).
    05  RD-ROW-TOTAL                 PIC Z(5)9.
```

Figure 3. Detail Print Line (Row)

Each of the examples illustrated in Figures 4 through 6 below shows the main features of a program segment that would print the table data. The first example (see Figure 4) shows the PERFORM...VARYING...UNTIL statement structure required under the COBOL-68/74 standards. In this case, the PERFORM statement forces the use of a

named procedure that contains the sequence of statements to be accomplished. As the iterative process used to fill the 10 column positions is accomplished within another iterative process that prints the 15 rows, two named procedures (one handling the filling of a row prior to printing and a second handling the printing of individual rows).

```
3100-PRINT-SUMMARY-REPORT.
     PERFORM 3110-PRINT-SUMMARY-HEADINGS.
     PERFORM 3120-PRINT-ROW-DETAIL
         VARYING ROW-INDEX FROM 1 BY 1
             UNTIL ROW-INDEX IS GREATER THAN 15.

3110-PRINT-SUMMARY-HEADINGS.
         :


3120-PRINT-ROW-DETAIL.
    MOVE WS-ROW-LABEL(ROW-INDEX) TO RD-ROW-LABEL.
    MOVE ZEROS TO WS-ROW-TOTAL-ACCUM.
    PERFORM 3121-MOVE-COLUMN-DATA-TO-ROW-DETAIL
        VARYING COLUMN-INDEX FROM 1 BY 1
            UNTIL COLUMN-INDEX IS GREATER THAN 10.
    MOVE WS-ROW-TOTAL-ACCUM TO RD-ROW-TOTAL.
    WRITE REPORT-RECORD FROM WS-ROW-DETAIL
       AFTER ADVANCING 2 LINES.


3121-MOVE-COLUMN-DATA-TO-ROW-DETAIL.
    SET COLUMN-OUT-INDEX TO COLUMN-INDEX.
    MOVE WS-DATA-ENTRY(ROW-INDEX,COLUMN-INDEX) TO
         RD-TABLE-ENTRY(COLUMN-OUT-INDEX).
    ADD  WS-DATA-ENTRY(ROW-INDEX,COLUMN-INDEX) TO
         WS-ROW-TOTAL-ACCUM.
```

Figure 4. Traditional (COBOL-68/74) coding

The second example (see Figure 5) illustrates the use of the in-line PERFORM feature of COBOL-85 where in-line, rather than named procedures, are used to control the iterative tasks. Consequently, the sequence of statements necessary to perform the desired tasks are physically located between the applicable PERFORM and its corresponding END-PERFORM.

```
3100-PRINT-SUMMARY-REPORT.
     PERFORM 3110-PRINT-SUMMARY-HEADINGS
     PERFORM VARYING ROW-INDEX FROM 1 BY 1
             UNTIL ROW-INDEX IS GREATER THAN 15
         MOVE WS-ROW-LABEL(ROW-INDEX) TO RD-ROW-LABEL
         MOVE ZEROS TO WS-ROW-TOTAL-ACCUM
         PERFORM VARYING COLUMN-INDEX FROM 1 BY 1
                 UNTIL COLUMN-INDEX IS GREATER THAN 10
             SET COLUMN-OUT-INDEX TO COLUMN-INDEX
             MOVE WS-DATA-ENTRY(ROW-INDEX,COLUMN-INDEX) TO
                  RD-TABLE-ENTRY(COLUMN-OUT-INDEX)
             ADD  WS-DATA-ENTRY(ROW-INDEX,COLUMN-INDEX) TO
                  WS-ROW-TOTAL-ACCUM
         END-PERFORM
         MOVE WS-ROW-TOTAL-ACCUM TO RD-ROW-TOTAL
         WRITE REPORT-RECORD FROM WS-ROW-DETAIL
             AFTER ADVANCING 2 LINES
     END-PERFORM.

3110-PRINT-SUMMARY-HEADINGS.
         :
```

Figure 5. In-line PERFORM

Because the printing of a row (i.e., a line on a report) suggests a task that may be viewed as a logical module, it seems reasonable that if named procedures are going to be considered for use, this task should be placed in one. On the other hand, when 10 of the 12 column positions in the output row were filled in using an iterative process, use of the iteration structure was simply one of convenience made possible by the tabular nature of the problem, and does not suggests the presence of a logical module. Following this line of reasoning, the third example (see Figure 6) illustrates a compromise approach in which a "named procedure" PERFORM is used to highlight the existence of a logical module (PRINT-ROW-DETAIL) that accomplishes the row printing process,with an in-line PERFORM used to indicate that the filling in of individual column positions in the output row is simply part of the row printing process, and not an activity significant enough to warrant a separate named procedure.

```
3100-PRINT-SUMMARY-REPORT.
     PERFORM 3110-PRINT-SUMMARY-HEADINGS
     PERFORM 3120-PRINT-ROW-DETAIL
         VARYING ROW-INDEX FROM 1 BY 1
             UNTIL ROW-INDEX IS GREATER THAN 15


3110-PRINT-SUMMARY-HEADINGS.
         :


3120-PRINT-ROW-DETAIL.
     MOVE WS-ROW-LABEL(ROW-INDEX) TO RD-ROW-LABEL
     MOVE ZEROS TO WS-ROW-TOTAL-ACCUM
     PERFORM VARYING COLUMN-INDEX FROM 1 BY 1
             UNTIL COLUMN-INDEX IS GREATER THAN 10
         SET COLUMN-OUT-INDEX TO COLUMN-INDEX
         MOVE WS-DATA-ENTRY(ROW-INDEX,COLUMN-INDEX) TO
              RD-TABLE-ENTRY(COLUMN-OUT-INDEX)
         ADD  WS-DATA-ENTRY(ROW-INDEX,COLUMN-INDEX) TO
              WS-ROW-TOTAL-ACCUM
    END-PERFORM
    MOVE WS-ROW-TOTAL-ACCUM TO RD-ROW-TOTAL
    WRITE REPORT-RECORD FROM WS-ROW-DETAIL
        AFTER ADVANCING 2 LINES.
```

Figure 6. A Compromise using both types of PERFORM

DISCUSSION AND CONCLUSIONS

The examples presented in this paper illustrate several important points. First, the COBOL-85 standard includes new features that provide significantly increased levels of power and flexibility to the professional business programmer. Secondly, this enhanced power and flexibility makes it possible to approach common programming tasks in a number of different ways. For instance, the in-line PERFORM could be used heavily, with wholesale elimination of separate named procedures, or it could be ignored completely. When and how it will be used will depend on the standards that are suggested or imposed during the next few years.

It is obvious from the examples presented in this paper that the development of guidelines regarding the use of the in-line PERFORM would be quite appropriate. One such guideline might be that when the activities to be accomplished within the span of control of a particular iteration structure exceeds 10 to 20 statements, those statements should be placed in a named procedure (or module) and executed within the context of the traditional form of the PERFORM, with task requiring fewer statements being accomplished under control of the in-line PERFORM. Another possible guideline might restrict in-line code to program control logic only, with

data manipulation code being placed in named procedures. The focus of these and other guidelines must, of course, be to ensure that this feature of COBOL-85 is used where appropriate to increase the reliability and maintainability of the code being generated, while reducing development costs through increased productivity.

While this discussion of various possible implementations of the PERFORM statement is instructive, it illustrates only one of many enhancements made to the language under COBOL-85. Implementation of the case structure with the EVALUATE statement, inclusion of a wide variety of scope terminators (i.e., END-PERFORM, END-READ, END-IF, etc.), addition of the ability to CALL...BY CONTENT/REFERENCE, and incorporation of many other such features has greatly increased the power and flexibility of COBOL. It is obvious that a great deal of thought, discussion and testing will be required in the next few years to determine how they can best be used to improve future COBOL code. A concerted effort must be undertaken by researchers, educators, textbook authors and professional business programmers to explore and identify COBOL-85 related programming techniques that, when implemented, will result in improved program readability, maintainability and accuracy as well as increased programmer productivity.

REFERENCES

1. Feingold, C and Wolff, L, Structured COBOL Programming (5ed), Wm. C. Brown Publishers, Dubuque, IA, 1988.

2. Fowler, GC, COBOL Structured Programming Techniques for Solving Problems, Boyd & Fraser, Boston, 1990.

3. Headrick, RW and Fowler, GC, *Structured programming techniques: a study of relative complexity*, International Journal of Computers & Education, 12(4):539-44, 1988.

4. Higgins, D, Designing Structured Programs, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

5. Marek, MF and Seabrook, RHC, Programming in C, Delmar Publishers, Inc., Albany, NY, 1989.

6. Yourdon, E, Techniques of Program Structure and Design, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1975.

TAU - a Relational Database System Laboratory

Michael Frame

Department of Computer Science and Information Systems
The American University
Washington, DC 20016

ABSTRACT

TAU is a relational database management system (RDBMS) created with the following goals in mind: (1) to build an RDBMS that could be used to demonstrate modern database concepts 'end users', programmers, and database administrators. (2) To develop a large software project that could be used by students to study the workings of RDBMSs. (3) To develop a basic RDBMS that could be used as a framework for RDBMS research and experimentation, or as a component of systems that require a database manager. (4) To provide a topic for software engineering projects. This paper explains the rationale for TAU and describes its structure and use in educational settings.

INTRODUCTION

This paper describes the rationale for and the structure of TAU, a relational database management system (RDBMS). Design and development of TAU addressed four major needs/issues:

* RDBMS's are needed that to demonstrate modern database concepts. TAU is to be used to instruct 'end users', programmers, and database administrators. For example, it can be used with Chapters 4, 5, 6, and 8 of Date [2].

* Advanced computer science students need access to the internals of RDBMS's in order to study the important algorithms, data structures, and design decisions that go into the construction of an RDBMS. TAU can support texts such as Ullman [5].

* A framework for doing RDBMS research or experimentation was needed. Our department plans a number of studies of RDBMS implementation issues that require access to the source code of a highly modular database manager. In addition, projects relating to user interfaces and database design require such a component.

* Projects in our software engineering courses need large realistic software in order to be able to construct realistic projects. An RDBMS has many components and interdependencies that can be used in such course projects.

Database systems are ideal for the study of computer science. They are an important topic in computing in their own right, but they also involve languages and their translators, optimization,

sorting, data structures, concurrency control, security, reliability, and integrity. In addition, 'front-ends' can be developed to illustrate human-computer interface problems.

TAU has been developed to be as portable as possible. The type and size of systems that might play host to TAU had to be considered. These considerations have had to be reflected in the architecture and code of TAU.

TAU is written in C and will eventually run in three environments: UNIX on the 3B2, IBM PCs, Apple Macintosh,. TAU can be a single user system on one of the smaller machines, or a multiuser system on minis. TAU is constructed so that it can be extended to run in various distributed environments. A number of projects are described to extend TAU for such environments.

TAU and its source code are available in machine readable form. It is our hope that others will want to use TAU and to implement it in new environments.

USING TAU

TAU is based upon the SQL language [1]. For most work users will be able to use the TAU ad hoc command processor, ITAU. This program allows a user to enter SQL commands and have them processed immediately. When a command is a query, the result is displayed to the user. The user has the ability to control some aspects of the format of the query output.

The TAU programmatic interface, PTAU is a library of entry points that may be used by C programmers to access TAU databases from their programs. ITAU is such a program. PTAU allows programmers to integrate TAU into their programs, or to write their own utility programs for TAU. TAU has the appearance of a subroutine library to C programmers.

The TAU database administrator utility, UTAU is the program that allows database administrators to create TAU databases, to reformat TAU databases, to save and restore TAU databases, and to monitor the use of TAU databases. Initially these tasks are performed largely with UNIX commands, but UTAU will grow to incorporate the necessary capabilities.

TAU INTERNALS OVERVIEW

The Sequential Database Model

TAU has a simple modular structure in order to be able to perform database experiments or to give assignments to students in database courses. In the first case, modularity isolates the object of the experiment. In the latter case, it helps eliminate needless complexity from student projects.

TAU is compose of the Translator, the Improver, and the Database Machine. Figure 1 illustrates the relationship of these components. TAU operates on SQL statements. These statements are submitted to the Translator which generates an intermediate language called Sequential Algebra. The output of the Translator is called a Sequential Algebra Expression (SAE).

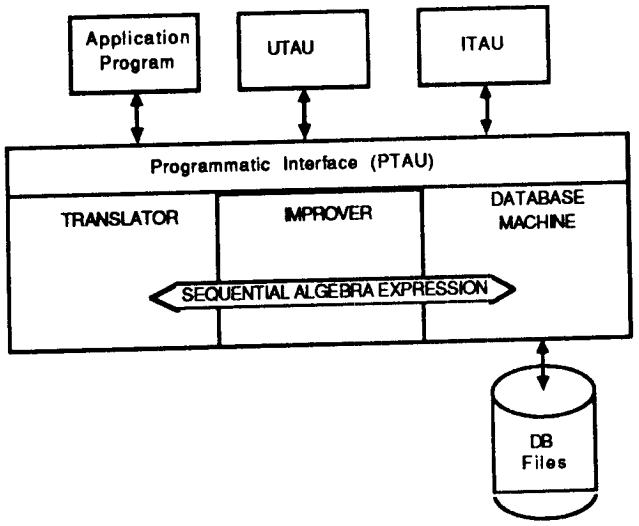


Figure 1. Architecture of TAU

The Translator does not attempt any optimization on the SQL statement, but only converts it into an SAE. The Improver processes the SAE generating a modified SAE that will be processed more efficiently by the Database Machine. Optimization is performed by the Improver alone. The Database Machine only carries out the steps specified by the SAE.

The Database Machine is an interpreter of SAE expressions. It fetches each SAE 'instruction' and invokes a corresponding procedure that performs the required operation. The operations call on other parts of the Database Machine which implement the access methods, and which also perform buffer management, logging, and recovery. The Database Machine implements the transaction concept which permits users to interact with the database without interfering with one another.

This structure has a number of benefits besides its simplicity. The decision to isolate the "intelligence" of the system in the Improver required us to implement the database machine so that it remained simple, but also so that improvements could be communicated to the Improver. In that way the Improver can always make a good decision. We describe each component more fully in the following sections.

The Tau Translator

The TAU translator is based upon attribute grammars [3]. An attribute grammar is a syntactic description of a language, but each production is extended to include semantic rules about that aspect of the language. The TAU attribute grammar for SQL has semantic rules that define how SQL statements are to be translated into Sequential Algebra Expressions. The result of SQL translation is a string of SA tokens (an SAE) that define how the SQL statement is to be interpreted.

A novel aspect of the Translator is that it can perform a translation by interpreting an attribute grammar, or it can generate a new translator (a C program) to efficiently translate a language. A designer can experiment with language syntax interpretively, then generate a 'hard-coded' translator for the chosen language.

The Tau Improver

An SAE generated by the Translator is not usually very efficient. Improver transforms an SAE into an equivalent form that can be interpreted more efficiently. Improver is the 'brains' of TAU.

The Improver is rule-based and the rules can be changed so that different combinations of optimizations can be tried. A rule is of the form

criteria: old-form -> new-form

where the criteria is a test to see if a particular condition is met. The 'old-form' is an implicit criterion in its own right. There has to be a pattern in the SAE that matches the 'old-form' before the rule can be applied. The criteria can test certain conditions that are recorded in the SAE by the translator, and they can also test dynamic information about the database or the database machine. For example, the Translator records information about which relations and attributes are referenced in each term in the SAE. The Improver could use this information in order to determine that a projection operator could be inserted to eliminate those attributes that are no longer needed in evaluating the result.

The information the Improver can get from the Database Machine includes: the number of rows in a relation, the existence of an index, how fast the Database Machine can perform a certain operation.

The Improver can perform not only formal (syntactic) improvements on an SAE, it can also perform cost-oriented improvements based upon the speed with which the database machine can perform its operations.

The Tau Database Machine

The Database Machine accepts an SAE and treats it in the same way that a CPU treats a machine language program. That is, each SA operator is an instruction to the Machine. The Machine is simple-minded and only carries out the orders it receives in the form of the SA string.

The Machine is composed of two major parts: the Interpreter and the Accessor. The Accessor is responsible for all operations to the physical database. The Accessor may be thought of as operating at the row (record) level, while the interpreter operates at the column (field) level.

It is possible to add new instructions to the machine. This involves simply adding an entry to an internal table and then implementing the corresponding procedure to carry out the operation. The interpreter has a driver that fetches each instruction and dispatches to the appropriate procedure. This modularity allows us to rewrite the procedures for existing operations. For example, we could devise a better join algorithm and simply replace the existing one. On the other hand, if we wanted to provide two types of join algorithms, we could add a new instruction type and corresponding procedure for the new algorithm.

Each operator has a number of information functions associated with it. When these functions are referenced, they return a value indicating how efficiently the Database Machine can perform the operation for the particular situation. Therefore, any time we change a procedure associated

with an operation, we must be sure to determine if the values returned by these functions need to be updated. When we add new instructions, we must determine how to set these function values.

The Accessor is based upon the concept of Frame Memory [4]. This is a design that provides a very flexible interface to secondary storage. The rest of the system has the impression that disk blocks are variable in size and can be stretched and shrunk as necessary. Another useful aspect of Frame Memory is that it was designed to make it possible to parameterize the design of the file structures and to easily calculate costs related to accessing operations. This is an important benefit for developing the cost values that the Database Machine must be able to provide to the Improver.

The Accessor is also modularized so that the buffer manager, the logging mechanism, and the locking and recovery mechanism are isolated from one another and from the rest of the system.

TEACHING WITH TAU

TAU is a complete RDBMS. Because of its ad hoc SQL interface, TAU can be used in database courses to teach SQL and database design. Students can experiment with SQL statements and with various logical and physical design problems. Advanced students could be assigned projects requiring them to develop database design aids based upon the current rule-base of the Improver. For example, knowing how the Improver will use indexes helps to determine which columns of which tables should be indexed in order for certain queries to be as efficient as possible.

Students have full access to the sources of TAU. They are able to implement improved algorithms for operations, to modify the access method, to make changes to the buffer manager, to add new data types, or to perform any number of other small or large projects. The advantage of using software like TAU is that the student can conduct a sizable one semester project and actually have something to show for it when the course is over.

EXPERIMENTS USING TAU

A number of types of experiments can be conducted with TAU. These include:

1. Implement a language other than SQL.
2. Experiment with changes to SQL syntax, for example implement the SQL2 syntax.
3. Experiment with changes to SQL semantics.
4. Try different rule sets.
5. Develop more efficient algorithms for evaluating rules.
6. Improve operations.
7. Add new operations.
8. Test exact cost values versus statistical cost values versus no cost values.
9. Experiment with differing parameters for the Frame Memory configuration.
10. Evaluate various buffer management algorithms described in the literature while holding the rest of the system constant.
11. Add new data types to TAU.

In addition, TAU can be extended in a number of ways:

1. Make TAU a distributed DBMS.
2. Implement a server version of TAU for LANs.
3. Use TAU as a component of a heterogeneous distributed database.
4. Make TAU user extensible.

Many of these topics are active areas of database research. Our hope is that TAU will make research possible for those who might not normally have the resources to explore these topics.

SUMMARY

TAU is a full function relational DBMS that is available in executable and in source code form. Instructors can use TAU to support database courses in both information systems and in computer science programs. The software can serve to demonstrate the uses of DBMS's to potential applications designers and database administrators. It can also demonstrate the internal workings of DBMS's to computer science students.

In addition, TAU can be used for conducting research on database topics relating to the uses of databases and database systems, and also to the implementation of database system software.

Finally, TAU is a set of programs that can be used in software engineering courses to support projects involving such topics as software maintenance, project planning and task time estimation, and group software development.

We hope to increase the number of users and supporters of TAU so that improvements in the product itself, and so that educational programs based on TAU, can be shared.

REFERENCES

[1] Database Language SQL, American National Standard X3.135-1986, American National Standard Institute, New York, NY, 1986.

[2] Date, C. J., An Introduction to Database Systems, Addison-Wesley, Reading, MA. 1986.

[3] Frame, Michael C. and Mehdi Owrang. "SQL Translation Using Attribute Grammar," submitted for publication.

[4] March, S. T., D. G. Severence and M. Wilens. "Frame Memory: A Storage Architecture to Support Rapid Design and Implementation of Efficient Databases," ACM Transactions on Database Systems, Vol. 6, No. 3, Sept. 1981, pp. 441-463.

[5] Ullman, J. D. Principles of Database Systems, 2nd Ed., Computer Science Press, Potomac, MD., 1982.

INTRODUCING DATABASE DESIGN USING OBJECTS

Thom Luce
Ohio University

ABSTRACT

This paper describes an experimental system which allows end-users to model objects in their environment and examine consequences of the chosen model. The system uses a series of pull-down menus and dialogue boxes to aid user development of the model and then uses a Prolog engine to "realize" objects in the model. Through cycles of modelling and object realization, end-users with no database training should be able to evolve reasonable database models.

INTRODUCTION

The word "object" has become one of the most overused and, perhaps, least understood words in the popular computer press today. When combined with the words "oriented" and "programming" it suggests an approach to programming that encapsulates knowledge about entities (objects) and the methods for processing them. Each object is an independent, modular structure which is able to respond to requests, called messages, from other objects.

Object oriented programming is also concerned with inheritance or the sharing of common attributes. One of the best known examples of inheritance is the "isa" relationship found in semantic networks[1]. According to this relationship, if an object called SPOT isa Dog and a DOG isa MAMMAL, then SPOT inherits the characteristics of dogs (four legs, tail, man's best friend, etc.) and the characteristics of mammals (warm blooded, hair/fur, milk producers, etc.).

A large number of special purpose languages have been developed to meet the need of object oriented programming. Included in this list are Smalltalk, C++, Actor, Flavors and others[2]. Object oriented systems have also been implemented directly in Lisp and Prolog[3,4].

Objects have also been used as the basis for systems analysis as described by Coad and Yourdon[5] and Shlaer and Mellor[6,7] and system design as indicated by Temte[8] and others. Booch describes object-oriented development "as an approach to software design in which the decomposition of a system is based on the concept of an object". He further suggests that "Rather than factoring our system into modules that denote operations, we instead structure our system around the objects that exist in our model of reality" (emphasis added).

Shlaer and Mellor consider objects as abstractions of real world things that are of interest to someone. Coad and Yourdon feel that the object paradigm is useful in systems analysis because it allows developers and users to communicate requirements in a natural framework of human organization, specifically the use of objects and attributes. They further believe that this emphasis forces developers to focus on the users world and application domain.

As pointed out by Kroenke[9], the words 'object oriented' are also used to describe database design. According to Peterson[10] the basic idea behind object oriented database design is similar to that of object oriented programming - entities in the real world can be

modeled with objects in a database.

Pinson and Wiener [11] point out that object oriented problem solving (of whatever flavor: programming, analysis, design, database design . . .) involves the identification of objects and what is done with them. During the initial stages of design users often perceive objects in terms of their principal attributes. Only later are keys, and relationships typically determined (Diederich and Milton [12]).

In his database book, Dave Kroenke [13] points out that users view their world as a collection of interrelated entities. A student does not, for example, look at a grade report and think of a table in third normal form representing permanent student information; a second table, also in third normal form, containing information on classes, and yet another table recording the fact that the student was enrolled in the class. The student does not think of and probably doesn't know that the enrollment table must contain a foreign key to relate it back to the student table and another foreign key to link the enrollment record with the appropriate class. Instead, the student sees the grade report as an entity, an object, containing various pieces of information related to her work for the quarter.

Viewing a problem as a series of objects can have some interesting results. For example, the dean of a college may view the departments under her management as objects containing, among other things, a department name, chairperson name and professors. The dean may also view the professors working for her as objects, each containing a professor name, office number and phone, and their department. The following figure shows the notation used by Kroenke to depict these objects. In this notation, boxes represent objects and the attributes associated with the objects are listed inside the box. A box inside a box shows an object contained inside another object (see Figure 1.).

While the relationship between departments and professors shown may make perfect sense to the dean, it can cause difficulty for the designer because of the infinite recursion implied by the nesting (departments contain professors which contain departments which contain . . .). There are, of course, common procedures available to the designer for resolution of this and similar problems. The nature of the relationship between the two objects can be determined (1:1, 1:M, M:N) and tables with the appropriate primary and foreign keys created and normalized. Procedures can then be developed for joining tables and selecting or projecting the appropriate information.

All this is well and good but where does it leave the user? The dean has no reason to know this process and shouldn't be expected to learn it. She does, however, need to know that her problem is understood and that her view of the relationship between the objects in her world will be retained. That concern is the focus of this paper.

A SYSTEM TO SUPPORT THE USE OF OBJECTS AS A DATABASE DESIGN TOOL

This paper reports continuing efforts to develop a database design aid based on the concept of objects as defined by Kroenke. The intent of this system is to allow users and designers to enter simple descriptions of objects along with sample data and determine if the resulting objects look like what the user intends. The system is written in Prolog and portions of the code have been described previously [14].

The first step in object oriented problem solving is identification of the objects in which the user is interested. It isn't necessary to identify all objects at once because new information can be added at any time. Following the example reported by Kroenke, a college dean might initially identify college,

department, professor and student objects in her world. These objects are entered by selecting "Draw objects" from the "Objects" pull down menu (see Figure 2.).

Objects are created by first moving the mouse cursor to the desired location on the screen and pressing the left mouse button which anchors the upper left hand corner of the object. Additional movement of the mouse causes a shadow of the object to be drawn on the screen. Once the desired shape is reached, pressing the right mouse button freezes the size of the object. The user is then asked for the name of the object and the shadow is replaced by a permanent object. Figure 3 shows the screen after defining college and department objects and drawing the shadow of a third object. Notice that the department object has a double border while the college object has only a single border. The double border indicates the currently selected, or active, object.

As indicated on the objects menu, objects may be moved, redrawn, or erased at any time.

In addition to objects, the user must identify some of the important characteristics of the objects. Attributes are entered for one object at a time. The object to be modified may be selected from a list of known objects or by pointing with the mouse. Figure 4 shows an object selection list and the menu choices used to access it.

Once an object has been selected, the "Attributes" menu selection brings up a dialogue box for entry of object attributes (see Figure 5.).

Notice that the user has defined Department as an attribute of college. As Kroenke points out, this is a very common occurrence. From the user's (the dean) point of view, departments are a part of the college. Once the attributes are entered and a key selected (optional at this time), the save button is selected and the system redraws the objects with their attributes. The fact that an object has been found inside another object is indicated by highlighting the object in reverse-video (see Figure 6.).

This process continues until attributes are entered for all objects. It isn't necessary to enter a complete list of attributes at this time because the list can be modified at a later time. Once the user is satisfied with the current model, sample data may be entered and the model tested. However, before this can happen the model must be transformed into a form that is easier to handle internally. This is accomplished with the "Transform definitions" selection from the "Process" menu (see Figure 7.). The transform definitions process asks the user to make some decisions related to objects found inside other objects. Each time an imbedded object is found the user sees a message as shown in Figure 8.

In this example the dean would select 'all' because all departments are contained inside the college. When asked about "Department" in student, the dean should pick specific to indicate the student's major (if that is what the department object inside student indicates). The choice for "Professor" inside student is more difficult and it is possible that neither choice will produce the desired result. This situation may indicate that something is missing in the model and point out the need for refinements.

Once data transformation is complete, sample data may be entered and edited for each object. This process requires that the user select the desired object and then pick "Data entry" from the Process menu. Data entry and editing use a dialogue screen as shown in Figure 9.

After entering sample data, the user can expand objects and examine the

consequences of design choices. This process, also known as object realization, is selected from the "Process" menu (see Figure 10.).

Object realization links and displays data that has been entered for an object, including information of contained objects. The example in Figure 11 shows a portion of the expanded college object after entering data on departments. At the time this expansion was performed no data had been entered for professors or students and hence the messages at the end of each department.

The student object contains both a department object and a professor object. If, during the transformation process, the user specifies that these objects refer to specific departments and professors then a expansion of object might look as shown in Figure 12.

Notice that all the basic information related to the professor and the department are shown just as the user envisions them when thinking about students. Notice also that the display does not mention the professor or student objects found inside department. This is also consistent with what the user most likely had in mind when thinking about students.

The process of object expansion may reveal problems with the objects that have been defined. Consider the simplified model of students and classes depicted in Figure 13.

This model indicates that student objects contain classes and class objects contain students. While this may be a perfectly normal and acceptable view of the world it causes problems for database management systems because it represents a many to many relationship. This problem is exposed to the user when the objects are realized as shown in Figure 14.

The data used for this example had only three students and two classes. Notice that every student is in every class and that every class contains all students. The problem, of course, is that the model is overly general. Student objects do not contain all class objects and class objects do not contain all student objects. Once this problem is recognized, the user may look for some way to alter the model to achieve the desired result. One solution might be the creation of a new object, perhaps called section, with students related to sections and classes related to sections. The modified model might then be as shown in Figure 15.

Experimentation with transformation of objects should show students in selected classes and classes with some, but not necessarily all, students.

SUMMARY

The system described in this paper is an attempt to allow users to design database systems by experimenting with the objects in their environment. Users can quickly enter objects, attributes and sample data and then examine realizations of their objects. Upon examination of the object realizations, it may be discovered that certain objects contain more, and possible incorrect, information than is desired. Other objects may lack necessary information. After noting these conditions, the user can easily modify the objects and their attributes and again examine the results. Following this procedure through a series of refinements, an acceptable model of the users environment can be developed for later conversion to a database management system.

FUTURE WORK AND RESEARCH OPPORTUNITIES

The object realization system reported in this paper is still under development and several areas require further work. Planned modifications will couple changes in object descriptions with changes in previously entered sample

data. Future versions will also allow users to convert the working model and sample data into a form that can be loaded into an actual database management system.

Improvements in object realization are also under consideration. It may, for example, be desirable to separate and dynamically control the limits on forward and backward linking. One additional area involves the differentiation Kroenke makes between multi-valued non-object attributes and multi-valued object attributes. The present system only recognizes multi-valued object attributes.

The author believes that an object realization system such as the one described here can be used in a number of research projects. Possible topics include studies on end user database definition and the potential role of object realization in the training of database management students. Studies comparing the quality of databases designed with the help of object realization to those designed following traditional techniques should also prove interesting.

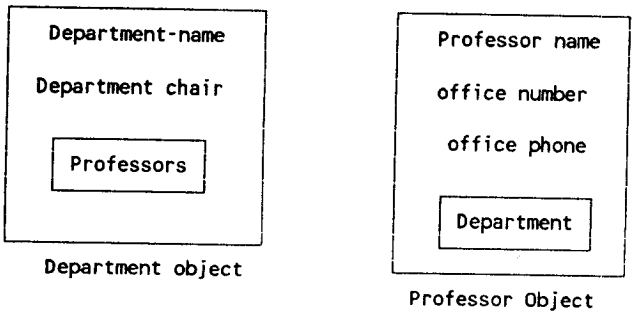


Figure 1. Examples of Objects

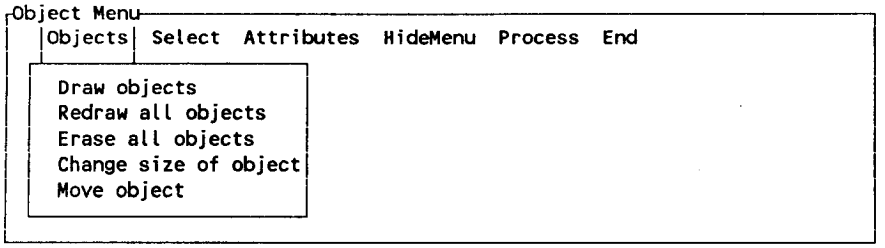


Figure 2. Objects Pull Down Menu

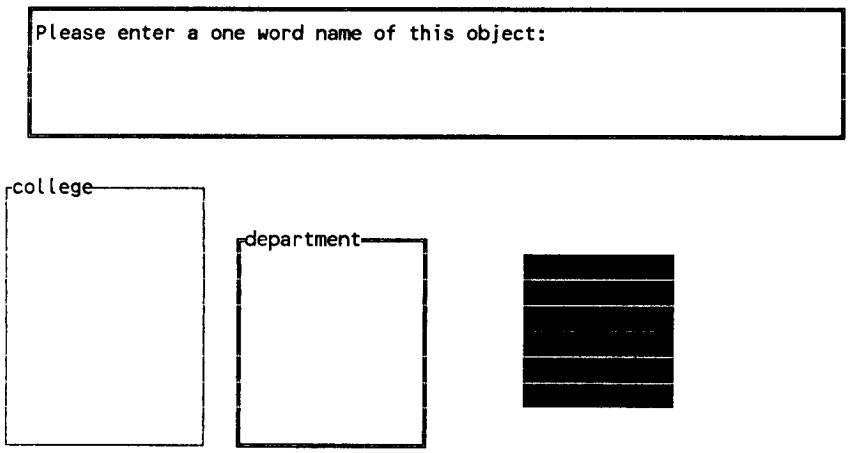


Figure 3. College Example

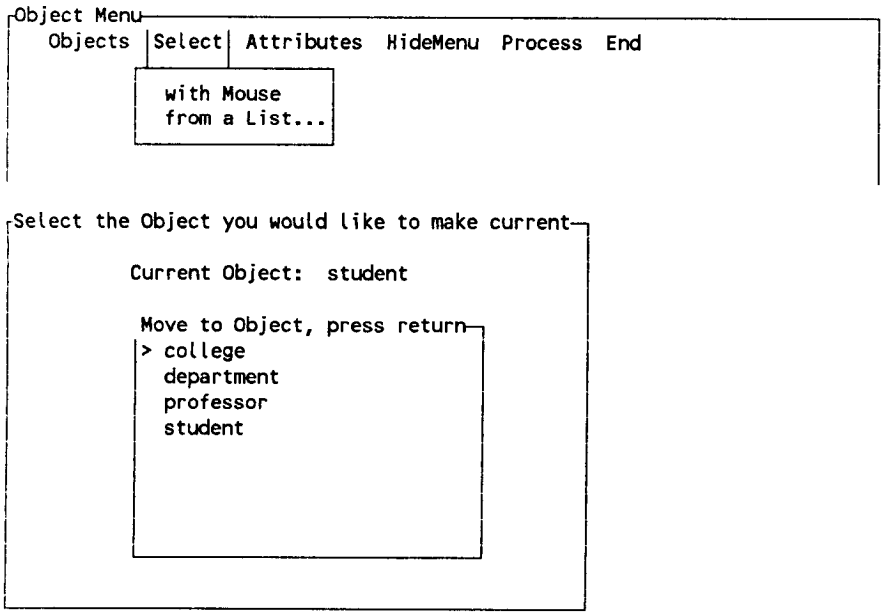


Figure 4. Object Selection List

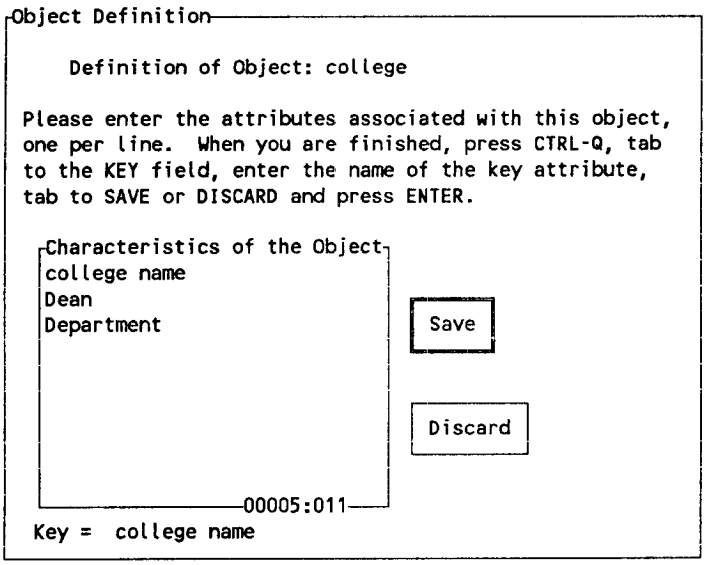


Figure 5. Object Definition

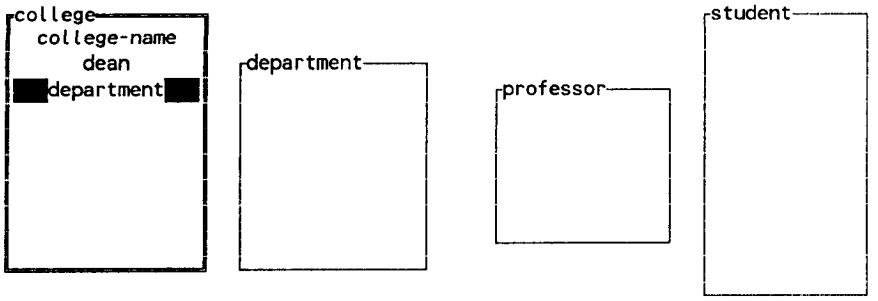


Figure 6. Relationship of Objects

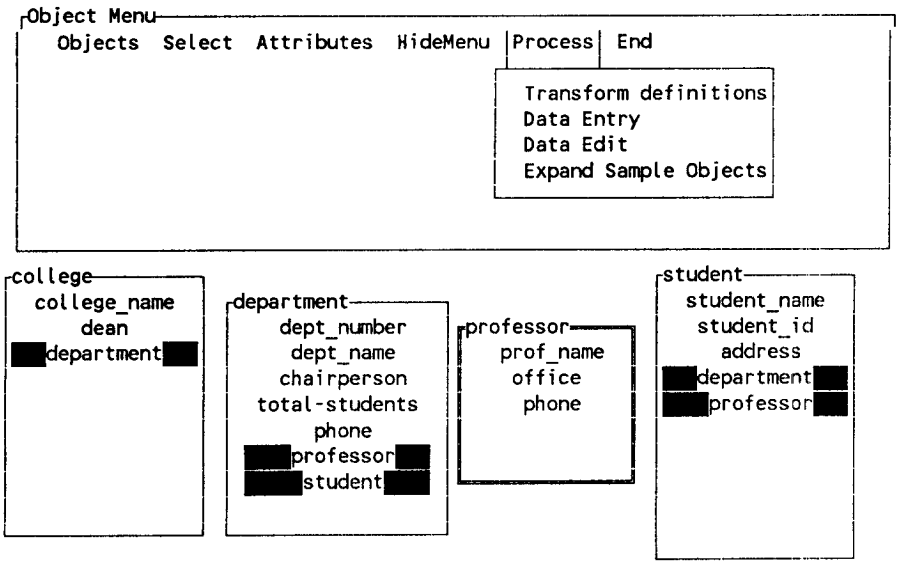


Figure 7. Transform Definitions

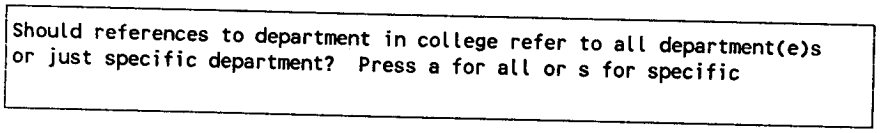


Figure 8. Transform Definition Option

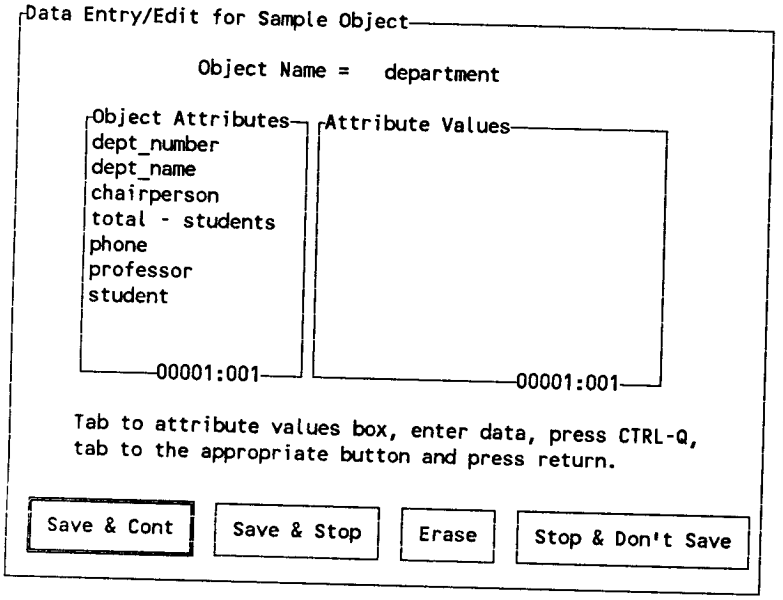


Figure 9. Data Entry/Edit Menu

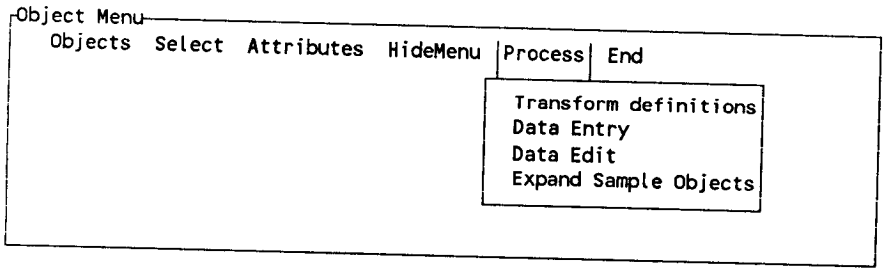


Figure 10. Process Menu

```
***college***
       college_name = Business Administration
       dean = William Day
       ***department***
              dept_number = 1
              dept_name = Management
              chairperson = Art Marinelli
              total - students = 550
              phone = 1111
              *** No professor(e)s located in database ***
              *** No student(e)s located in database ***
       ***department***
              dept_number = 2
              dept_name = accounting
              chairperson = Ted Compton
              total - students = 400
              phone = 2222
              *** No professor(e)s located in database ***
              *** No student(e)s located in database ***
       ***department***
              dept_number = 3
              dept_name = finance

                                Press ENTER to continue
```

Figure 11. Expansion of "College" Object

```
***student***
       student_name = I.M.A. Student
       student_id = 111
       address = Athens OH
       ***department***
              dept_number = 1
              dept_name = Management
              chairperson = Art Marinelli
              total - students = 550
              phone = 1111
       ***professor***
              prof_name = Thom Luce
              office = Copeland 1F
              phone = 2058
```

Figure 12. Expansion of "Student" Object

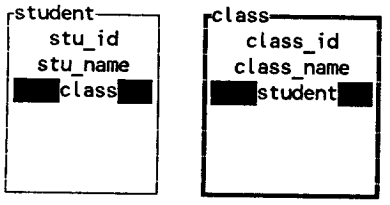


Figure 13. Many to Many Relationship

```
(expansion of class)
 ***class***
        class_id = a
        class_name = class a
        ***student***
                stu_id = 1
                stu_name = one
        ***student***
                stu_id = 2
                stu_name = two
        ***student***
                stu_id = 3
                stu_name = three
***class***
        class_id = b
        class_name = class b
        ***student***
                stu_id = 1
                stu_name = one
        ***student***
                stu_id = 2
                stu_name = two
        ***student***
                stu_id = 3
                stu_name = three
```

```
(expansion of student)
 ***student***
        stu_id = 1
        stu_name = one
        ***class***
                class_id = a
                class_name = class a
        ***class***
                class_id = b
                class_name = class b
***student***
        stu_id = 2
        stu_name = two
        ***class***
                class_id = a
                class_name = class a
        ***class***
                class_id = b
                class_name = class b
***student***
        stu_id = 3
        stu_name = three
        ***class***
                class_id = a
                class_name = class a
        ***class***
                class_id = b
                class_name = class b
```

Figure 14. Expansion of "Class" and "Student"

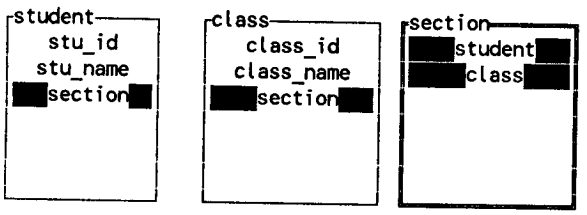


Figure 15. Expansion of Many to Many Relationship

REFERENCES

1.Waterman, Donald A, A Guide to Expert Systems. Addison-Wesley Publishing Co., 1896, pg 70-71.

2.Rettig, Marc, Tom Morgan, Jeff Jacobs, Doug Wimberly, "Object-Oriented Programming in AI: New Choices", AI Expert January 1989, pp 53-69.

3.Stabler, Edward, P. "Object-Oriented Programming in Prolog", AI Expert, Oct 1986, pp 47-57.

4.Newton, Mike and John Watkins, "The Combination of Logic and Objects for Knowledge Representation" Journal of Object-Oriented Programming 1(4) Nov/Dec 1988, pp 7-9.

5.Coad, Peter and Edward Yourdon, Object-Oriented Analysis, Prentice-Hall, Inc. 1990.

6.Shlaer, Sally and Stephen J. Mellor, Object-Oriented Systems Analysis, Yourdon Press 1988.

7.Temte, Mark, "Object-Oriented Design and Ballistic Software", ACM Ada Letters, 4(3) November/December 1984, pp 25-36.

8.Booch, Grady, "Object-Oriented Development", IEEE Transactions on Software Engineering, Vol SE-12(2), February 1986, pages 211-221.

9.Kroenke, David M. "Teaching Database Processing with a Object Orientation", SRA MIS Newsletter, 1(1) Fall 1987.

10.Peterson, Robert W. "Object-Oriented Data Base Design", AI Expert, March 1987, pp 27-31.

11.Pinson, Lewis J. and Richard S. Wiener, An Introduction to Object-Oriented Programming and Smalltalk, Addison-Wesley Publishing Co., 1988.

12.Diederich, Jim and Jack Milton "Objects, Messages, and Rules in Database Design", in Object-Oriented Concepts, Database, and Applications, edited by Won Kim and Frederick H. Lochovsky, Addison-Wesley, Publishing Co., 1989, pages 177 - 198.

13.Kroenke, David M. and Kathleen A. Dolan, Database Processing: Fundamentals, Design, Implementation, Science Research Associates, Inc., 1988.

14.Luce, Thomas G. "Object Oriented Design with Prolog", Proceedings of the 17th Annual North American Conference of the International Business Schools Computer Users Group, pages 215 - 228.

A Practical Approach to the Database Management Systems Course

Hossein Saiedian
Mathematics and Computer Science
University of Nebraska

ABSTRACT

A one-semester Database Management Systems course has become quite common in both the Computer Science and Information Systems curriculums. The purpose of this paper is to share with the academic faculty of these two disciplines a project-oriented approach used by the author to teach such a course. In the course described in this paper, the theoretical concepts of databases are presented in the classroom and the students are requested to apply these concepts during the design process of a database project assigned to them. Students form teams, and each team implements and maintains a database for a small business enterprise. The database design life cycle used by student teams as well as the description of what is expected of each team is presented.

INTRODUCTION

The author has taught, and is presently teaching a project-oriented Database Management Systems (DBMS) course. This course is a one-semester course taught twice a year and has been intended for senior/first-year graduate students majoring in Computer Science or Information Systems.

A course in DBMS is an important part of a Computer Science or Information System curriculum, not only because of the theoretical aspects of DBMS's, but also because of the needs of the industry and the business community. A study by Archer [Archer83] shows that business and industry consider DBMS as one of the most important courses in a list of common Computer Science or Information Science courses offered.

There are various approaches to teaching a DBMS course in colleges and universities. Some instructors discuss the theory of the databases only while others may discuss the theory, design and implementation concepts but only in the traditional form of class-lectures. The author's approach has been a theory-practice approach. While the theoretical concepts of the DBMS's are presented in the class, the students are also requested to form database design teams and to apply the concepts during the design phases of the database project that is given to them during the first week of the semester. That is, the students will apply the learned concepts while actually implementing and maintaining a database for a small business enterprise. The students' response to this approach has generally been very positive because they could "feel" the results of their understandings of the concepts. The purpose of this paper is to reminisce the approach that author has employed in hopes that it might be useful for those who are currently teaching a DBMS course or to those who are planning to teach such a course in future.

PROJECT CHOICE

During the first week of the classes, the students are asked to form 4-member database design teams. Each team elects a team leader. The team leader has the responsibility of coordinating the activities of the other team members as well as communicating with the instructor.

choices as explained below:

1. A 'Project Description' describing a problem situation is given to the students by the instructor. The project description describes the operations of a small enterprise and asks students to consider designing a database that would address the information needs (e.g., information retrieval, storage, update; report-generation, etc.) of that enterprise. (The choice of enterprise changes every semester. Currently we are considering a manufacturing company.)

2. The students are asked to go into the business community and actually pick a local business to design a database for. To ensure that the database design process does not go out of control, the students are warned not to choose a large business.

Both of the above two choices have their own advantages and disadvantages. Choice 1 has the sole advantage that the students have the project description in their hands and do not have to make any trips outside the campus. It has the disadvantage that the team members very often have to make assumptions about the operations, problems and needs of the enterprise. Depending on the assumptions made, the design process may be oversimplified, unnecessarily made too difficult, or inconsistence.

Choice 2 has the advantage that the students will have to deal with a real world situation and will realize how important it is to have good communication skills to interact with users who often are ambiguous and unclear as to what they want. Thus the students will gain important real-world experience while interacting with their client. The obvious disadvantage of this choice is that the team members have to make trips to outside campus during the design process. (It must be mentioned that some local businesses are very receptive of the students. In fact, I was called by the manager of a local business last Fall who asked me to send a team to investigate the feasibility of designing a database for his business the next time this course was to be offered.)

Nevertheless, the choices are made by the teams. The members of those teams that have chosen the instructor's project will have to play the roles both the client as well as the designer. Usually, the teams that design a database for an actual business end up being more satisfied with their end-product because of its realistic nature.

COURSE FORMAT

The intent of this course is to present a broad overview of the most important concepts of databases and is organized in the traditional manner with the instructor presenting the material. Students should have had a data structure class before taking this course.

The course begins with an introduction of the basic terminology and concepts, file processing vs. database processing, advantages of using a database, the ideas of data models and data modeling techniques, and typical architecture of most database management systems. A discussion of importance of using a high-level data model is then provided and the basic concepts of Entity-Relationship model [Chen76] (ER) as well as ER diagramming technique are presented.

A database design life cycle is provided which serves as basis for the database design project. This life cycle is introduced below. The rest of the course is taught more or less the "standard" way: a discussion and comparison of major implementation models (i.e., the relational, network and the hierarchical models), as well as database languages. The author makes

heavy uses of the relational model and languages. The main issues facing the database designers are then presented which includes a long discussion of the theory of functional dependencies, and the normalization techniques. Other database concepts and issues such as backup and recovery, concurrent access to the database, security, advances in database modeling (e.g., object-oriented databases) and distributed database systems are discussed later in the semester.

DATABASE PROJECT DESIGN LIFE CYCLE

The database design process is an "art." Generally speaking, it can be defined as the process of examining the requirements and building a conceptual schema that is a model of the business. The conceptual schema is then mapped into a logical model which in turn is mapped to physical structures.

The students are provided with a life cycle which they follow to design their databases. This life cycle has five major phases. A simplified version of the phases of this life cycle is briefly reviewed below:

1. **Predesign Evaluation/Requirements Analysis Phase**

   a) Organizational Survey: The team members study the operations of the business, identify the shortcoming and problems, determine what functions are performed, who performs the functions, what forms and documents are used and what procedures are followed.

   b) Feasibility Study: The potentials for using a database are investigated. It is determined if the database will be cost-effective.

   c) Functional Analysis: The functions performed as well their input/output are specified. Who performs the function, how frequent, etc., are also documented.

Tools and techniques used during this phase include: Questionnaires, interviews, observations, HIPO charts, data flow diagrams, Warnier-Orr diagrams [Orr81] and SADT [Ross77].

2. **Conceptual Schema Design Phase**

   a) Data Collection/Data Dictionary: Properties of data elements that affect the database, e.g., type, format, size, etc. are documented. Data analysis is also done at this point and the relationship among the data (i.e., 1-1, 1-n and n-m relationships) are also documented. The end result is a data dictionary.

   b) Conceptual Schema Design: Entities are formed and an ER diagram is produced. It is emphasized that ER modeling is the key to the development of database. By using the basic constructs of ER model, the conceptual schema for the enterprise is established. This conceptual schema is used for clarifying communications with various people interested in the database being built and for developing improved approaches for the database design. Furthermore, the conceptual schema establishes a basis for speculating about the long-term changes in the database organization.

3. **DBMS Selection Phase**

   A discussion of important factors that must be considered when selecting a DBMS such as suitability, user-interface, maintenance, and efficiency as well as the technical and economical issues are presented. For the class

project, a relational DBMS is usually selected. Currently we are planning to use DEC VAX Rdb database management system software.

4. **Logical Model Mapping Phase**

   The conceptual schema (represented in terms of ER diagrams) is mapped to the logical data model of the underlying database management system. For this course, the conceptual schema is mapped into relational database schemas. Algorithms for mapping ER diagrams to relational schemas (as well as to network and hierarchical schemas) are given to the students.

5. **Database Implementation Phase**

   The database is implemented. Criteria for implementation (e.g., response time, space utilization, etc.) are discussed and guidelines for implementing the system are provided. Importance of good user interface is discussed and students are recommended to provide good user interface for their database project. Some teams provide menu-driven and screen-oriented interfaces while others provide a query-driven interface. The motel database mentioned earlier provides a graphical interface.

Steps 1-4 are referred to as logical database design process while step 5 is referred to as physical database design process.

STUDENTS' PROJECT ASSIGNMENTS

Each team is responsible for designing a database for the project they have chosen. To provide feed back for the students, projects are divided into six parts, numbered one through six. Each project part will be turned in twice --- once when it is initially created and once at the end of the semester as parts of the completed project. The instructor grades each project part and returns it to the teams along with an evaluation sheet. The evaluation sheet shows the grade for that project part and includes correction remarks, comments, and/or suggestions for improving the project. Students are told that they may receive the lost points back if necessary corrections are made when they turn in their completed project. The following is a brief description of what each team turns in throughout the semester:

1. **Project Part 1: Report of Predesign Evaluation/Functional Specification**

   This part includes the following:

   a) A report describing the system requirements to be incorporated into the database. In addition, this report will serve as a basis for mutual understanding between the designers (i.e., students) and their clients.

   b) A description of problem areas and recommendations for correcting these areas as well information on performance requirements, and preliminary design features for creating the database.

   c) The costs involved in converting to a DBMS. The students are asked to make reasonable assumptions in stating the costs.

   d) A statement concerning the feasibility of implementing the proposed system. (The students are told that it really does not matter whether their conclusion is that the system is feasible or not, we will proceed.)

   e) A set of functional specifications describing the major functions performed in the business, including the input and output of these functions and the frequency of use, users, and other relevant information about

each function.

2. **Project Part 2: Data Dictionary**

The initial data dictionary for the database is turned in. Although in practice a comprehensive data dictionary contains data attributes, relations, schemas, subschemas, and reports, for students' projects we settle for the data attributes only. (Relational schemas are turned in as a separate document.) The data dictionary will of course grow in entries over the semester. Common attributes for each data item of data dictionary include: data name, aliases, data type, format, range, availability, security, dependencies, and comments. The data dictionary will serve as an integral tool throughout the design process.

3. **Project Part 3: The Conceptual Schema (ER Diagram)**

Based on the outputs of phases one and two, each team creates a conceptual schema using the ER model. The ER diagram most generally will change slightly as the students better understand the project. Students are encouraged to employ the enhanced features of ER model when diagramming.

4. **Project Part 4: Data Dependencies**

For this part, the students are required to discover and document all dependencies between data attributes. The dependencies are normally determined in conversations between students and their clients. Students are also asked to make reasonable assumptions about data dependencies and clearly document their assumptions.

5. **Project Part 5: Relational Schemas**

For this part, the students create a relational database schema from the ER diagram as well as using the data dependencies of the previous project part. Once the relational schemas are determined, the relations are normalized to achieve the desired normal form, usually the Boyce-Code Normal Form (BCNF). Students are taught and are asked to use both the decomposition technique [Ullman82] as well as the synthesis algorithm [Bernstein76] and compare the result of the two. Since the output of the synthesis algorithm may not be in BCNF, some teams use this algorithm to generate Third-Normal Form (3NF) schemas and then use the decomposition technique to achieve BCNF.

For each relation schema, students define the attributes, functional dependencies (FD's), multivalued dependencies (MVD's) if any, integrity constraints, candidate key(s), and foreign key(s). Normally, each database has around 8--10 relation schemas.

6. **Project Part 6: Implementation**

The students first create relational schemas in the chosen DBMS and then implement the database by creating empty tables, and then adding data. Students test their database to make sure that they can update (i.e., inset, delete, and modify) tuples, and query the database for relevant information. Students must enforce key, entity, and referential integrity constraints. Key integrity constraint means that the primary key of each relation must always have unique value. In other words, no two tuples in a relation should be allowed to have equal key values. Entity integrity constraints means that no attribute participating in a primary key should be allowed to have null values while referential integrity constraint implies that foreign key attributes should have values that match the value of the base relation or else should have

null values.

At this point, each team must have completed its project. Teams are scheduled to demonstrate their database.

In addition to the above, each team is also required to do a class presentation. Presentations are formal and are given to the entire class and consist of a description of the structure of the database, and the team's approach to designing it. The presentations are usually centered around the ER diagram. Instructor as well as other students are allowed to ask questions.

Each part of a project is graded based on accuracy and completeness of its content as well as its organization (e.g., appropriate title, section and paragraph names) and appearance (e.g., consistent page numbers). Each graded part is given back to the students who are supposed to make necessary corrections and modifications. At the end of the semester, all project parts are put together as the final version of the project. The final project is once again graded for completeness and consistency. Forty to fifty percent of a student's final grade is based on the final project.

CONCLUSION

It has been my aim to provide the database students with a "real-world" and hands-on experience in a database course. The students are exposed to the challenges of working in teams to develop a database to address the information needs of a small business enterprise. The development of this course has been a real learning experience for me and I hope that this paper will provide a few suggestions to those who are or will be teaching a DBMS course.

REFERENCES

1. [Archer83] Archer, C.B.: "What Does Business and Industry Expects from Computer Science Graduates Today?," ACM SIGCSE Bulletin, Vol. 15(1), (Feb. 1983), pp. 82-83.

2. [Bernstein76] Bernstein, P.: "Synthesizing Third Normal Form from Functional Dependencies," ACM Trans. on Database Systems, Vol 1(4), (Dec. 1976).

3. [Chen76] Chen, P.: "The Entity-Relationship Model --- Toward a Unified View of Database," ACM Trans. Database Systems, Vol. 1(1), (March 1976), pp. 9-37.

4. [Orr81] Orr, K.: Structured Requirements Definition, Ken Orr and Associates, 1981.

5. [Ross77] Ross, D.: "SADT," IEEE Trans. Software Engineering, Vol. 3(1), (Jan. 1977).

6. [Ullman82] Ullman, J.: Principle of Database Systems, Second Edition, Computer Science Press, 1982.

Computer Literacy for Business Persons in Six Easy Lessons

Ronald J. MacKinnon
Mathematics and Computing Sciences Department
St. Francis Xavier University
Antigonish, Nova Scotia B2G 1C0

ABSTRACT

It is essential for most business persons to be microcomputer literate in order to successfully compete in today's business marketplace. The availability of inexpensive micros and micro software has meant that virtually any business can afford a micro. However, it has been a major problem to educate mature business persons about the fundamentals of micros and their appropriate use. This paper will explain a six lesson micro literacy course that has been taught by the author many times over the past seven years. In this paper the contents of each lesson and the method of teaching will be explained. The software found to be most appropriate will be identified and the criteria for selecting the software will be discussed. Every offering of this course was evaluated and some of the results of this evaluation will be discussed. The author's thirty years of experience in the computer field as a manager, teacher and as a computer consultant for small business has provided the background for selecting the material taught and the teaching approach used.

OVERVIEW

The author has been teaching adults, especially teachers about computers for over twenty years and has been teaching with and about micros since the development of microcomputers in 1978. The micro literacy short course described in this paper has been taught and refined over a period of seven years and has been taught and evaluated many times during this period.

The course has six lessons because it has been the authors experience that many adults get tired and bored if there are more classes. After taking six classes many students in the course evaluation often recommend additional classes. Yet if additional classes are added many students then complain that the course is too long. An additional reason for the six class length is that the instructor can complete a good overview of micro fundamentals, word processing, spreadsheets, file management, integrated software and with instruction in the BASIC language in that period of time.

There should be one micro for each student to use. Some administrators propose two persons per micro in order to have more students use a limited number of micros (and thus more profit, if there is a charge for this course), but as expected, the student does not learn nearly as well when he/she is sharing a computer with someone. The author has taught semester long courses with the "hands on" approach but with two students sharing a micro and has seen the contrast with the results of having one person using one micro.

The outline of the micro literacy course is shown in Table 1. The contents of each lesson along with the rational for

choosing this material will now be explained. Since the software used is crucial to the success for the course, the choice of software will also be explained.

LESSON 1

Lesson 1 is the most important lesson in the course. All adults have a great fear of computers but they are not aware that all the other members of the class have the same fear. It is very important at the beginning of the first class to inform the class of the universality of this fear and that they are not alone in their concerns. Once this is accomplished, the fundamental terms concerning micros (hardware, software, cpu, operating system, memory, storage and formatting) should be explained in simple terms. It is important that these terms be explained in the simplest terms possible. If complex words are used to explain these terms they will not be understood by the students and often this will turn off students from the course and they will either quit the course or do a mediocre job with the rest of the course. Humor and personal stories often are a great help here to set the students at ease.

A few fundamental commands of MSDOS; DIR, DIR/W, DIR/P, FORMAT A:, FORMAT A:/S, FORMAT A:/V, FORMAT A:/S/V, DISKCOPY and COPY should be explained and demonstrated by "hands on" use of the computer. It is important that students actually use each of these commands on their micro. It is not sufficient for the students to receive a lecture from the instructor where they are passive observers, even if the instructor demonstrates the commands. This is the hardest part of the course for some persons and they have a need to be reassured that the course will not get more difficult and that they will get an opportunity to review these commands. One technique that reinforces the fundamental MSDOS commands is to give the students some time to run a MSDOS shareware tutorial disk. The students find that the pressure to perform at the same speed as the class is removed and they enjoy working at a speed that is comfortable for them and is controlled by them. The definition of what shareware software means should be explained, then the student can use the DISKCOPY or COPY commands to make a copy of the shareware diskette and can be assured that this process is perfectly legal and a bonus for the student is to let the student keep the copy of the diskette. Everyone likes to get something free and this is a great motivator to successfully duplicate the diskette. This diskette also has the additional benefit that the student can run the software at their own leisure at home or the office. This is a nice ending to the first class where the student successfully duplicates a diskette. The author has used several MSDOS tutorial software products both commercial and shareware over the past few years and has found that the TUTOR.COM shareware program by Computer Knowledge has a friendly explanation of the keyboard, hardware and fundamental DOS commands. Students have enjoyed using this software.

It is important to ask the students to bring a letter or some work they would like to print out with the word processor next week. The students should be assured that even though they have never used a computer before, it is guaranteed that they will type up and print out some material at the next class. The student then has something to look forward to and is anxious to achieve this success at the next class. This simple promise also ensures that the students come to the next class because there is still a lingering fear among many adults that this computer material might be too difficult for them.

LESSON 2

In lesson two, it is helpful to review

the material covered in lesson one. This reinforcement helps make adults feel more at ease with computers and their knowledge of them. If possible, incorporate some humor in this review. It breaks the tension and students feel more comfortable and think that perhaps this course will not be as bad as they feared.

Word processing is the major theme of lesson two. It is important to use a very user friendly word processor for this lesson. Although WordPerfect and Microsoft Word are two superb word processors, they are not appropriate choices for a word processor for this lesson. PC-Write is also a very good value as a word processor and is a shareware program but it is also not a good choice for a word processor for this workshop because of its relatively poor ease of learning and ease of use. In recent ratings of word processors in Infoworld, the first choice as an executive word processor was Professional Write which also had the best rating for ease of learning and ease of use among word processors. Having taught several word processors, it has been the author's experience that Professional Write is the easiest word processor to teach and students have success quicker and easier with this word processor then any other. An alternative word processor would be Q & A which will also provide an excellent file manager for use in Lesson 4.

It is a good practice to keep it simple when showing adults how to use the word processor. Hands on use with one micro per person is crucial for word processing. The students should learn how to correct a mistake, delete a letter and insert a letter or group of letters. The basic word processing functions of retrieving, saving and printing a file should be taught as soon as convenient. Centering a line, underlining and boldfacing words are also easy to teach and convenient to use. Students are especially impressed with the use of the speller. It is helpful to have a sample document with spelling errors in it, so that students can retrieve the document and check the spelling. There is great satisfaction for the students to successfully use the speller. It is important to have the students type up a short letter or report and print it out. Like little children in elementary school, these adult students are delighted to bring home to a spouse or a friend something that they have done on a computer and have printed out themselves. It has been the author's experience that the printing out of this document is the key to success for this word processing unit.

LESSON 3

Although the word processing lesson is likely the easiest lesson for the student and immediately useful, the third lesson on spreadsheets is likely the most useful software that a business person can learn. After teaching many spreadsheets over several years, it has been the author's experience that the Lotus type spreadsheet, in addition to being the industry standard, is also sufficiently easy to learn. There are several good shareware spreadsheets that operate like Lotus 1-2-3. In particular, the shareware spreadsheet "As Easy As" works like Lotus, can read Lotus files and has a major benefit that this disk can be legally duplicated and given to the student. Again the meaning of shareware should be explained to the student, so that they can register the software if they should decide to use it. Spreadsheets are easily taught by loading a simple spreadsheet and having the student move the cursor between cells to see the contents of the cell. It is extremely helpful to load a large spreadsheet and make several changes to it, so that the students can see how all the formulas are recalculated. The immediate reaction of most bookkeepers and accountants is that they will not be able to live without a spreadsheet and

they usually say "Why didn't someone tell me how easy it is to use computers!". Most students like to create a simple spreadsheet by themselves. It is useful to give the students a simple spreadsheet with a summation formula in it and have them enter the data as a class step by step. Students also like to have a significantly larger spreadsheet that will be a challenge to them. They get great satisfaction in completing such a spreadsheet by themselves.

Occasionally there are a group of students that can complete all the spreadsheet work with little difficulty. With this kind of a class, they really enjoy learning how to create an autoexec.bat file and making a selfbooting diskette that will immediately load and run "As Easy As" or some other software.

LESSON 4

In Lesson 4 the principle of a database and the extraction of information from the database via a file manager should be explained. Students often are not aware that the usual way to get information from a computer was to write a program. Again the selection of software is crucial. The critical factors in choosing a file manager are ease of learning and ease of learning, the same criteria as used in choosing a word processor. There are many file managers on the market but two stand out in the areas of user friendly, Q & A and Professional File. Students who have used Professional Write find out that they feel that they already know Professional File because the interface is so similar. Actually Q & A and Professional File are so similar, students can learn one program and they find that they are also competent with the second package.

In this lesson, the concepts can be easily explained by example. By loading a database, searching the database and adding data to an existing database and preparing a simple report, students can learn very quickly about file management. Students enjoy creating their own databases. For adult students, the creation of a database that is useful in their business or work is very helpful.

LESSON 5

Lesson 5 is one that the author has experimented with many variations. Although it may seem illogical, many adults have a desire to write a program in a computer language. It can be explained that spreadsheets can provide them with more useful information, quicker and easier, many adults seem to have a need to learn the fundamentals of BASIC. Having taught BASIC for many years to undergraduates and children, it is a constant amazement to the author that adults are as excited as children with their success in writing BASIC programs. Although there is not a good practical reason for teaching BASIC in a six lesson micro short course, it is included in the micro literacy course by popular demand.

Students can be shown the fundamentals of BASIC and learn how to list, run, save and print the programs. Some students are satisfied with attempting to write several programs for problems given to them. However, other students get satisfaction in running a tutorial program about BASIC. There is a shareware tutorial program on BASIC called PC Professor that is widely available and is popular with students.

LESSON 6

The last lesson is the wrap up of the course. One of the factors students look forward to is the reception of their official certificate stating that they have successfully completed a 12 hour course on the applications of micros in business. The importance of an official certificate should not be

overlooked. This certificate is often the deciding factor in some students choosing to take this course and is certainly a factor in students remaining in the course to the finish.

In this last lesson, the use of an integrated software package like Pfs:First Choice or Microsoft Works is a nice way to complete the course. Although Microsoft works is an excellent package, the author has found that after using Pro Write and Pro File, students find they already know Pfs:First Choice. Students are interested in learning a new software package and they are delighted to find out how easy it is to learn this new package. From their experiences in this course, students also learn to check out any software made by Software Publishing Corporation. Actually, many students think the company is called Pfs. It is also very useful to have feedback from the course participants concerning all aspects of the course. The length of the course, the content of the course, what subjects should be dropped or added and the instructors performance should be continuously evaluated. The content of this short course was decided after evaluating this short course offered several times a year over a seven year period. Students also like the opportunity to work on a project of their own choosing during this last class in order to get the assistance of the instructor.

CONCLUSION

The above described course gives a student a well rounded introduction to the use of micros in business. The course could be longer and cover more topics or cover the above topics in more depth. However after trying out all of these suggestions made by the students, the course outlined above has proven to be the best combination of topics and students have continuously rated the course very highly and leave the course wanting more instruction about micros and often enroll in more advanced courses about spreadsheets, database management systems, word processing or accounting. It is far better to have students leaving this course wanting more instruction than having them leave the course saying the course was long and boring.

TABLE 1

COURSE OUTLINE OF APPLICATIONS OF MICROS FOR BUSINESS

Lesson 1. Micro fundamentals and MSDOS

Explanation and example of: Hardware, Software, CPU, Operating System,Formatting.
DIR, DIR/W, DIR/P
FORMAT A:, FORMAT A:/S, FORMAT A:/S/V
DISKCOPY, COPY.
Have students copy IBM Tutor disk, they keep it.
Students run tutorial on MSDOS, IBM Tutor.
Tell students to bring a letter to type next week.

Lesson 2. Word Processing

Review of Lesson 1
Pfs: Professional Write
Fundamentals; save, retrieve and print.
Type up letter, save letter, check spelling and print it.

Lesson 3. Spreadsheets.

As Easy As Spreadsheet, Lotus compatible spreadsheet
Format disk, copy As Easy As disk, students keep copy.
Look at easy spreadsheet
Do simple spreadsheet.
Do HBUDGET spreadsheet.
If appropriate, create autoexec.bat.

Lesson 4. File Management

Pfs: Professional File
Go over fundamentals.
Look at several small databases.
Search databases.
Add to database.
Create database and add info to it.

Lesson 5. BASIC Language

Fundamentals of BASIC.
Write simple programs, list, save and print them.
Run PC Professor, shareware tutorial on BASIC.

Lesson 6. Integrated software and wrap up

Hand out certificates.
Course evaluation.
Pfs: First Choice integrated software.
Open period to work on any topic.

IS Professionals Plugging /\ into IS Education

Eli Cohen
Bradley University

ABSTRACT

This paper describes a four step teaching pedagogy, based in teaching theory and research, that is successful for teaching a technical course to non-technically minded students. It has been found successful when used in a three hour evening course when students typically are not at peak attention. The four steps are 1) reading by students, 2) lecture by an IS professional on a topic related to the professional's job, followed by a lecture by the professor filling in topics not covered in the guest lecture, 3) practice solving business problems using case studies, and 4) analysis and synthesis of a solution to a real business problem as a term project. This pedagogy engages students through all their modalities and confronts students at all levels of the cognitive domain.

The purpose of this paper is to share with other Information System Educators a pedagogy for teaching technical courses that enhances the course's relevance and excitement. The pedagogy is particularly effective for classes taught one evening a week. It involves four steps:

1) reading about a new concept,
2) hearing about it both from an IS professional and from the professor,
3) doing an in-class group case activity, and
4) creating a case study by analyzing a business.

What is new about this pedagogy is using this activities in combination.

Theory of Learning and Teaching

Studies on teaching and learning demonstrate that certain pedagogical elements will enhance learning. Two of these pedagogical elements are present in the pedagogy described below.

1. Bloom et al. taxonomized the cognitive domain of educational objectives into six levels: knowledge, comprehension, application, analysis, synthesis, and evaluation. The knowledge objective refers to such activities as being able to recount a specific fact. Comprehension refers to activities such as restating a principle in one's own words. Application includes activities such as applying a principle to a new situation. Analysis includes pointing out unstated assumptions. Synthesis calls for producing a plan or reorganizing ideas. Evaluation calls for the application of informed judgement based on some criteria or evidence.

The importance of this is that we as teachers need to teach not just for memorization of facts (the knowledge level), but also for the other five levels in understanding.

2. Students learn through various modalities: hearing, seeing, and doing. Some students by nature will learn better through one modality and others through another. To reach all students,

all modalities should be employed.

Problem in Teaching Technical Course to the non-technical student.

As MIS professors know, DPMA's model curriculum for CIS recommends that colleges teach, as part of their IS curriculum, courses that are challenging to teach due to their technical nature. For example, the Distributed Computing course is a required technical course for the MIS major at this university. Our students are not accustomed to technical subject matter and so the course has developed a reputation among students and faculty alike as being a hard course. To make matters worse, the course is scheduled at times as an evening class, from 6:00 to 9:00 at night. Students taking evening classes are often tired; taking a class for three hours regardless of the hour is by itself tiring. Add to this the technical nature of this course and the class can become both dreary and tedious.

After reviewing the literature on teaching described above, I synthesized a pedagogy suitable for teaching a technical course to non-technically minded students during a three hour stretch in the evening. This pedagogy draws the students to learn through four interdependent activities--Reading about the topic, Hearing about the topic, Doing a class exercise, and Creating and Presenting a case. These four activities are described below:

1. Students read in their text and handouts about the topic of the day. The students tend to keep up on the reading since they know that the following two activities that require the reading will be taking place that evening.

2. An IS professional addresses the class on the topic of the evening or a portion of it. These professionals are selected because they work daily in the field about which they will talk. In addition, after the speaker has completed, I lecture on the material covered in the readings, but not covered by the speaker.

3. The students break into groups to solve business problems using the information learned in this chapter.

4. During the course of the semester, each student analyzes one of the cooperating businesses in terms of its telecommunication-distributed processing needs and develops a case based on their analysis and synthesis.

This pedagogy draws heavily on the cooperation of IS professionals in the community, and thus "plugs IS professionals into IS education". It utilizes all the learning modalities; students learn through seeing, hearing, speaking, and writing. The students see and hear the speakers, speak as part of the in-class solving business problems and when presenting their case, and write their cases.

It also uses all the levels of educational objectives: knowledge and comprehension are addressed in the students' reading of the text and hearing of the lecture, application and analysis of knowledge are addressed by the visiting IS professional, and synthesis and evaluation are addressed by the in-class solving of business problems and by developing a case.

These steps are now described in more detail.

Reading

As in the typical class, students read the text assignment for the class. Occasionally students are required to read additional matter not covered in the text in preparation for the

evening's topic; the field changes so quickly that no text is complete.

Speakers

IS Professionals deliver the first hour's lecture. The list of speakers and topics is shown below as Table 1. As you see, the program of speakers progresses from non-technical to technical topics as the students in that class become more technically competent. Also the topics progress roughly in parallel with the text. The IS professionals are drawn from local businesses (although a few drive over an hour to share their time with the class). During the second hour of class, I lecture on areas of the evening's topic not covered in the IS professional's guest lecture.

Solve Case Study Business Problems

During the last hour of each class, students form groups to solve problems assigned them that night regarding the case study businesses. These assignments apply the knowledge they learned on the current topic of the course. These case studies are unusual in that they deal with three types of business situations:

1. a small, single site plumbing company,
2. a small chain of auto repair shops, and
3. a large, multi-layer conglomerate.

During the first class period, I provide the students written case studies of these business situations. The cases demonstrate businesses that need to develop distributed processing solutions and indeed need to communicate one with the other.

Each case assignment requires the groups of students to develop a technical solution for each of the three firms, thereby gaining a better understanding of how business needs differ for different firms. That is, the groups apply what they have learned in class that day to these cases. At the end of class, each group reports its solution to the case and is critiqued by the other groups. This critiquing adds excitement as the groups compete for the best solutions.

Student Developed Case Studies

Students are required to develop a term paper or analyze the telecommunications needs of a local firm as a term project. By conducting such an analysis, students need to analyze and synthesize, thereby consolidating their understanding of the topics studied in class. But, even more, having the students work in the community with IS professionals helps plug our students into IS, and plug IS professionals into IS education.

For the final days of the course, the students play the role of IS professional, giving lectures on what they learned while developing their term paper or project. In this way, the students giving the lecture practice speaking skills as well as educating their fellow students.

Summary

This paper describes a four step teaching pedagogy, based in teaching theory and research, that is successful for teaching a technical course to non-technically minded students. It has been found successful when used in a three hour evening course when students typically are not at peak attention. The four steps are 1) reading by students, 2) lecture by an IS professional on a topic related to the professional's job, followed by a lecture by the professor filling in topics not covered in the guest lecture, 3) practice solving business problems using case studies, and 4) analysis and synthesis of a solution to a real business problem as a term project. This pedagogy engages students through

all their modalities and confronts students at all levels of the cognitive domain.

BIBLIOGRAPHY

Bloom, B.S., Engelhart, M.D., Furst, E.J. Hill, W.H. and Krathwohl, D.R. (eds.), Taxonomy of Educational Objectives: The classification of Educational Goals, Handbook I: Cognitive Domain. David McKay, New York: 1956.

DeCecco, John P. and Wm. R. Crawford. The Psychology of Learning and Instruction, 2nd ed. Prentice Hall, Englewood Cliffs, NJ: 1974.

DPMA Model Curriculum for Undergraduate Computer Information Systems Education. Education Foundation, Data Processing Management Association, Park Ridge, IL: 1985.

Dunkin, Michael J. and Bruce J. Biddle. The Study of Teaching. Holt, Rinehart and Winston, Inc. New York: 1974.

Hyman, Ronald T. Ways of Teaching, 2nd ed. Harper & Row, New York. 1974.

Table 1: Speakers' Topics and Class Periods

Class #	Topic
2	Careers in Telecommunications and DP
3	Fiber Optics
4	Tech. Issues re: adding the Freenet bulletin board
5	The Library Network System
6	Novell Network Architecture and LANs
7	University's Campus Network
8	(tour of University Campus Network)
9	Internet
10	(local company's) Transnational Network
11	ISDN
12	Students as IS professionals and teachers
13	Student Presentations on their Cases
14	Student Presentations on their Cases

INDIVIDUALIZED COMPUTER LITERACY COURSES: WHY AND HOW THIS IDEA CAN BE SUCCESSFUL

Mary Lynn Manns
Computer Science Department
University of North Carolina at Asheville

ABSTRACT

Computer literacy classes often contain students with diverse aptitudes, backgrounds, and interests. This paper addresses a university which has successfully created more homogeneous computer literacy courses and, as a result, is offering new and attractive opportunities to both the students and the faculty.

In the Computer Science Department at the University of North Carolina at Asheville an effort is being made to allow undergraduate computer literacy courses to contain a more homogeneous group of students and at the same time, offer more challenging opportunities to both the students and the instructors. This endeavor has been successful through the creation of unique computer literacy courses offered to students who possess specified academic qualities. The evolving plan has consisted of three courses, as follows: 1) a version of an introductory BASIC programming course for students exploring a major in computer science, 2) a section of the microcomputer software course for students in the university Honors Program, and 3) an accelerated version of the microcomputer software course offered only to computer science upperclassman. Each of these three courses is described in detail in the following paragraphs.

The first course, CSci 161, was a special section of the department's service course in BASIC programming. (Since the demand for BASIC programming course sections has dropped, CSci 161 is no longer offered at UNCA, but while the demand for BASIC was high, it was a successful addition to the department's program.) It enrolled students who may have not yet decided upon a definite major but were strongly considering computer science. These students were provided with more challenging programming assignments covering much more material than the standard course in BASIC programming. Unlike the students in the standard BASIC classes who were working towards fulfilling a cognate requirement for another major, the students in CSci 161 **needed** a more demanding syllabus in order to gain a strong stepping stone to the first **required** course in the university's computer science curriculum.

The second course, CSci 173, presently offers a special opportunity for students in the university's Honors Program. Just as in the standard microcomputer software course, CSci 126, the Honors version starts with introductory principals. But because of the distinctive nature of the students, CSci 173 is able to cover many more advanced principals and do so at a much faster pace. Most students enter the course with word processing knowledge; those few that don't have this experience attend workshops on campus and are tutored individually. DOS, spreadsheet, and database packages are covered. The students then examine topics which are not part of CSci 126, including packages such as expert systems and Macintosh graphics. Because these students tend to be self-starters who achieve success by working on their own, the instructor teaches the core

material and then allows the method of evaluation to depend more on individual projects than on universal exercises and exams. While the students are working on their projects, special instruction can be given to those who have less experience or desire advanced information about a certain concept. The Honors Program students are often involved in a variety of undergraduate research at the university. The individual projects offer the opportunity to use their newly acquired computer skills to enhance their research. Projects are presented in a well-documented manner and, through formal presentation, are discussed with the class. As the instructor, I am constantly impressed and have thoroughly enjoyed helping students advance beyond the core material presented in class lecture to the point of creating unique and sophisticated projects.

Before the third course, Micro Software Applications for Programmers, CSci 351, was created, computer science upperclassman often entered the introductory, microcomputer software packages course, CSci 126. They did learn something about the packages but, in the process, were often bored with the slow pace of this introductory computer literacy course. Since a great deal of the computer science course work at UNCA is done in a timesharing VAX environment, many upperclassman find a need for a micro software packages course which is similar to CSci 126 but they desire one that is conducted at a more attractive level and pace. This upperclassman version of CSci 126 does not need to spend time with basic computer literacy terms and techniques. Rather, it allows students who are approaching graduation to spend the time surveying current microcomputer software on the market. Topics such as micro operating systems, spreadsheets, databases, IBM and MAC graphics, and expert systems are studied in relative detail. Class includes lectures, but students are also required to research materials such as manuals and other documentation in anticipation of what will be expected of them in almost any computer career. The primary method of evaluation is individual projects, which are often completed for campus users. Students are expected to prepare and present these projects in a professional manner.

Although these three exact courses may not be appropriate for the university reading this paper, I would like to encourage faculty to consider this idea of individualizing computer literacy courses in their curriculum. The students appreciate these courses because classes which normally have a very diverse population of students can now contain students who are at a much similar level of interest and aptitude. Faculty also value this idea because it creates many distinct opportunities to test challenging ideas, allowing for uncommon classroom projects at the undergraduate level. It is for these important reasons that the UNCA Computer Science department will probably continue introducing more individualized courses in the future.

THE USE OF A NETWORK SIMULATION PACKAGE IN A COMPUTER NETWORKS COURSE

Curt M. White
Department of Computer Science
Indiana-Purdue University at Fort Wayne

ABSTRACT

Data communications/computer network courses often include one or more lectures on local area network design. Many times an assignment is given to create a hypothetical network for a given problem. Unless the school has a laboratory with multiple machines and interconnection hardware, most assignments stop at the design phase. A relatively inexpensive alternative to network creation is to perform a network simulation using a modern network simulation package. This paper describes the use of one such package in a university computer networks course.

INTRODUCTION

Many colleges and universities offer some type of data communications/computer networks course in which the basic concepts of local area networks are introduced. Typically the course instructor covers the common topologies: bus, ring, and star. While it is common to give the students the necessary facts such as throughput, number of processors, collision rates, etc., it would be more beneficial if the students were required to create their own. The initial step would be a thorough analysis and design of the existing situation using paper and pencil. Unless the school has the resources to actually assemble a local area network as designed [1], this is often where the assignment ends. A reasonable intermediate step which may not be prohibitively expensive is to have the students perform a simulation of their designed network using a modern network simulator. This paper will examine the level of detail necessary to supply the simulation package the appropriate parameters, followed by the pitfalls and positive points of a past experience.

BACKGROUND

The Computer Science Department at Indiana U.-Purdue U. Fort Wayne offers two courses in the area of data communications. The first course is an introduction to the basic concepts of data communications, including data transmission and encoding, data link control, circuit and packet switching, radio and satellite networks, local area networks, and an introduction to OSI layered architecture. The second course, Computer Networks, deals extensively with the communication architecture of OSI and various examples of existing networks. In this course an assignment is given to design a local area network for four stations on a conveyor belt system. Included with the four stations is a processor/file server which gathers data and prints reports. The students are asked to consider two topologies: a collision bus and a token bus.

The initial step of the assignment is for each student to devise a rough layout of the network considering the number of processing elements and their instruction sets, the number, location and parameters of any storage devices, the necessary software modules, and the

necessary files residing on storage devices.

The next step of the assignment requires the students to present their solutions to the class for group discussion. From this interaction the students discover strengths and weaknesses of their suggested solutions. After any modifications, the students prepare the required parameters for performing a network simulation using a commercial network simulator. As we will see in the next section, the number of parameters necessary to properly run a network simulation requires the student to more deeply examine and understand exactly what they are trying to achieve and how much is involved in suggesting a LAN solution. Through this deeper understanding the student learns further the complexities involved in LAN design.

Previous studies involving network simulation in the classroom have been limited. There are examples of early network modelling in the classroom using Simula [2], and numerous examples of network modelling in the industrial sector, but very little has been published concerning the use of advanced network simulators in the classroom.

THE SIMULATOR

The simulator used for the course is NETWORK II.5 by CACI Products Company.* To create a simulation one begins by defining five basic units: statistical distribution functions, processing elements, busses, storage devices, and software components. Processing elements model any hardware device which is not merely a data sink or source. Examples include sensors, bus controllers, a cpu, or a personal computer. A processing element is further defined by:

- basic cycle time,
- time slice,
- interrupt overhead,
- an input controller,
- and the set of instructions.

This set of instructions lists the possible instructions that may execute on a particular processing element, and may include Processing, Read, Write, Message, and Semaphore instructions. Each type of instruction is further defined by a particular set of parameters dependent upon instruction type. For example, a Read instruction would need to know which storage device and file to access, the number of bits to transmit, and on which bus to transmit the data.

A bus is defined by:

- cycle time,
- bits per cycle,
- words per block,
- word overhead time,
- block overhead time,
- protocol (FCFS, collision, priority, token ring and slotted token ring),
- and the list of processing elements and storage devices.

In addition, each protocol requires defining a separate set of parameters, dependent upon the protocol chosen.

A storage device is defined by:

- word access time,
- bits per word,
- words per block,
- overhead time per block access,
- capacity,
- and the number of ports.

The software components consist of four basic types: modules, instruction mixes, macro instructions, and files.

The most common type of module is one that specifies a task which is to be performed by a processing element. To define a module one includes:

- module interruptability,
- absence or presence of concurrent execution,
- iteration period,
- start time,

* CACI is based in San Diego, CA. Phone is (619) 457-9681.

- set of processors on which this module is allowed to execute,
- set of preconditions which must exist before this module will execute,
- list of instructions this module is to execute,
- and the successor module(s) which follows execution of this module.

An instruction mix is a "pseudo-instruction" which is a list of names of instructions, other instruction mixes, and macro instructions with a percentage associated with each.

A macro instruction is, as its name implies, a collection of instructions which are referenced by a single name, while a file is defined by its name, its capacity, the name of the storage device on which this file exists, and an option of read-only status.

The Network II.5 simulator contains more features but these are the essential items necessary to model a "simple" local area network. Clearly, the student must possess a greater understanding of the problem environment and its intended solution than simply recommending we "hook all the computers to a bus and put a file server on it". By having the students model their designed local area network, they are forced to examine in much greater detail items such as equipment specifications and capacities, user requirements, environmental characteristics, and software requirements.

PITFALLS

Since a very small percentage of students were computer engineering technology students, most had difficulty finding the more technical information for their models. Values such as processor/bus/disk cycle times and word/block overhead times had to be given to students for fear that inaccurate times would generate invalid modeling results. Even when values were given many students felt uncomfortable that if one particular parameter value was off by 10 ms, an entirely different set of results could be obtained.

Students also had some difficulty with the concept that at one point one defines the set of instructions a processing element can perform, then at a later time defines the software modules which use these instructions.

A third area of difficulty was, when selecting a collision type bus, determining the values for collision window interval, retry interval, and contention interval. While it is common to encounter these terms in lecture, it is more difficult for a student to determine actual values.

Several students complained of the learning curve necessary to input and operate a model using Network II.5, while others noticed that the learning curve was nothing unusual for such a powerful simulator.

Another problem area was long compile and execute times. Although the simulator is designed to operate on a variety of machines, both micro and mini, our lab consisted only of 8088 and 80286 compatible microcomputers.

A final problem area that many students encountered was a simulator error of concurrent execution. Due to the complexity of software modules consisting of individual instructions operating on specific hardware elements with specific cycle times, many times a module would try to repeat execution before it had finished. This problem became more acute when software modules/hardware devices were queued due to bus collisions. Too often the student simply changed the parameter to allow concurrent execution. Unfortunately, this only got rid of the error message, not the reason for the error. What was necessary was a more detailed examination of all clock times involved.

POSITIVE POINTS

Many students commented that what they learned might have "real world" application some day. More than just talking about networks, they were able to create their own and then see if what they created would work in a simulation. One student commented that the best part of the project was actually seeing how every piece fit together to make a complete system work.

Other comments consisted of attaining a greater appreciation for computer modeling and designing local area networks; gaining the knowledge necessary to allow multiple processors to work simultaneously while limiting bus collisions; and a better understanding of the steps necessary for solving other large problems.

CONCLUSIONS

Most felt that the exercise greatly expanded their outlook on what was necessary to properly design a local area network. While the assignment was kept as simple as possible, it was still a learning process for both instructor and students. Next semester we will attempt a more difficult design, or have the students work in groups of two or three, each group trying at least two different designs and comparing their results with each other. It is clear however that a tool such as a network simulator has vast possibilities for in-course instruction and does not have to be limited to local area networks, but could include long-haul networks, or even multiple processor computer designs (3).

REFERENCES

(1) Hughes, L., 'Low-cost networks and gateways for teaching data communications', ACM SIGCSE Bulletin, Vol.21, No.1, Feb 89.

(2) McCoy, J.M., French, S.L., Abnous, R., Niccolai, M.J., 'A local computer network simulation', ACM SIGCSE Bulletin, Vol.13, No.1, Feb 81.

(3) Thuente, D.J., 'Shared memory multiprocessor simulation techniques with Network II.5', Proceedings Society for Computer Simulation Western Multiconference, San Diego, CA, Jan 90.

THE IMPERATIVE FOR A NEW APPROACH TO TELECOMMUNICATIONS EDUCATION FOR THE INFORMATION SYSTEMS PROFESSIONAL

Julian W. Riehl
Virginia Commonwealth University

ABSTRACT

This paper develops the rationale for a definitive approach to telecommunications education -- one that specifically addresses the knowledge and skill requirements of the information systems professional. It describes a two-semester program of telecommunications education developed within a university undergraduate information systems curriculum to meet this growing demand. The success of the program plus a number of distinctive features -- such as its case-study oriented approach -- that set it apart from more traditional approaches to telecommunications education serve to recommend it as a model for similar curriculum development undertakings.

INTRODUCTION

The information systems profession has been in a continuous state of metamorphis since the introduction of the electronic digital computer into business-type organizations in the early 1950's. In recent years telecommunications technology has added an entirely new dimension to the challenges faced by information systems specialists struggling to maintain professional competency in their protean craft.

Today, data communications provides the critical electronic infrastructure which all major corporations use to tie together their complex networks of processes, data resources, and diverse operating units. As such, data communications has become an integral part of computer-based information systems.

One significant result of this dramatic reordering of the national and global communication systems has been a revolution in the design of information systems, as well. In the modern business world distributed processing, on-line interactive information systems have become the order of the day. In their more fashionable forms they further assume the more esoteric characteristics of local area networks, cooperative processing and client/server architectures.

The impact of these developments in information systems design has been to force an increasing degree of integration of the technologies of information processing and information transfer (data communications). As a result the information systems professionals -- the analysts, designers, project leaders -- often find themselves in the uncomfortable position of assuming responsibity for the development of an application system whose ultimate success depends on a technology -- data communications -- of which they typically will have little or no understanding.

PROBLEMS OF SELF-EDUCATION

Unfortunately, acquiring an effective level of data communications competency has proved to be an extremely difficult and frustrating process for the individual, for a number of reasons:

-- Until recently the typical college information systems curriculum did not include courses in telecommunications. Even today with growing interest in telecommunications education the academic attention is largely centered on four-year engineering or computer science type curricula that are designed to produce data communications technical specialists. [7][12][13]

-- Telecommunications proves to be an extremely daunting subject for self-study. The underlying technical detail is extremely complex and arcane with a seemingly infinite variety of operational processes involved. Furthermore, there is little in common betweeen the more familiar technology of developing computer application programs, which concerns itself largely with controlling the processes within the computer, and that of data communications which deals with the electrical transfer of data between computers.

-- The process of self-education is further complicated by the unrelenting and accelerating pace of the technology and its applications.

The requirement for a practical degree of data communications competency on the part of the information systems professional is made even more critical because of the growing recognition that telecommunications can be a key strategic factor in the business planning of the firm. A number of writers have documented this trend; among them are Keen [5], Porter [8], Wiseman [14].

EDUCATION PROGRAM DEVELOPMENT

In recognition of the clear imperative for telecommunications education tailored to the needs of the information systems professional, the Information Systems Department at Virginia Commonwealth University (V.C.U.) undertook the development of a specialized program of study in telecommunications within the undergraduate information systems curriculum.

The cornerstone of the V.C.U course development effort was the generation of a comprehensive set of knowledge requirements that would meet the needs of the information systems work-place. This was accomplished through a university-industry cooperative effort involving a field survey conducted on-site at eleven major corporate users and providers of telecommunication services. The companies represented a variety of sizes and business orientations: both regional and national firms in manufacturing, retail, banking, insurance, electrical power distribution industries plus software and system integration vendors were involved.

Research Results

The results of the field research confirmed the need for information systems professional-oriented telecommunications education and established a number of basic criteria and concepts to guide the actual educational program design:

1. The basic objectives of the course of instruction should be to emphasize fundamental concepts of the technology and develop a general proficiency on the part of the student in the principles and processes involved in the design of data communications networks from the unique perspective of the information systems analyst, designer or project leader.

2. The need to integrate the telecommunications support

requirements into the basic information systems development process should be an underlying theme of the instruction.

3. Some means of allowing the student to apply the basic fundamentals of telecommunications to real-world design situations should be provided. Case study problem-solving exercises are considered to be the best classroom method of accomplishing this.

4. The student should be provided with a working knowledge of the wide spectrum of applications of telecommunications technology that exists. The history of the technology and governmental regulatory processes that shape it should be understood. This for the purpose of interpreting the the major trends, current and future, and their anticipated impact on information systems design.

5. The ultimate basic program of instruction should consist of not more than two semester-long courses. This is because of the already over-crowded undergraduate information systems curriculum and the practical limitations of the study time that the average working professional will have available.

Program Implementation Principles

The above guidelines have been implemented in a two-course sequence within the undergraduate information systems curriculum at Virginia Commonwealth University. The first course focuses on the basic principles and concepts of data communications; the follow-on course builds upon these fundamentals and applies them to practical network design situations.

The successful implementation of the program served to validate several principles related to course presentation and organization. They are summarized below:

1. Because of the specialized target audience for the program -- the entry-level and working information systems professional -- the course content, and more specifically, its emphasis and selection is critical to the effectiveness of the instruction. This problem is covered further in item 2., below.

2. The typical basic telecommunications text suffers from a number of inadequancies, when considered in light of the curriculum objectives. These inadequacies apply to the following areas: selectivity in the presentation of technical detail, providing an information systems design orientation throughout, integration of the technologies of telecommunications and computer-based systems into a common methodology at all levels of information systems planning and design; and most importantly of all, offering the student a problem-solving environment for integrating the technical fundamentals with actual applications of the technology.

3. The need to integrate the learning experience with real-world problem solving is probably the most critical element needed to provide the most effective possible learning evironment. Both the research and the implementation experience clearly indicate that the best way of achieving this goal is through the use of case studies. Several introductory texts offer case examples, e.g., Black [1], Stallings [9], Stamper [11]; none of them, however, offers problem-solving within the broad framework of a case study.

4. Case studies, while a central element of the education model, however, must be used with some caution. Experience has shown that introducing case

studies too early in the basic course causes the student to spend a disproportionate amount of time on the mechanics of problem solving at the expense of absorbing the technical fundamentals. For an introductory course the most effective point of application of the case study appears to be in its second half. For the follow-on course in data communications networks the case study approach has been found to be an ideal framework around which the entire curriculum can be designed. Significant details of the case studies developed for the V.C.U. telecommunications program are discussed below.

CASE STUDY DESCRIPTIONS

The introductory course at V.C.U. employs a single case study used during the last four weeks of a fifteen week course. The case study involves a multi-drop network in support of a on-line business transaction system. This particular application environment was selected, after considerable experimentation, because it offers a good vehicle for summarizing the students' learning of wide area network principles and provides the opportunity to compare the relative performance characteristics of a variety of network components and configurations.
The case study further presents the opportunity for the introduction to queueing-based mathematical modelling, the comparison of design alternatives within a cost-benefit analysis framework and offers the student a working example of close interaction between information system processes and network design.

The second course in the V.C.U. undergraduate telecommunications program is one that builds on the basics of the first course and applies them, in an introductory fashion, to a number of real-world network design situations. The centerpiece of the course is a continuing case study, which provides the focus for study activity throughout and exercises the student in basic design applications in both wide-area and local area networks.

The case study, which is framed in a retail business environment, is divided into three major segments: 1) decentralized, switched line networks; 2) interactive, on-line systems support; 3) local area networks. In the first segment, the students study switched-line tariffs and design options including advanced dial-up modem technology and the use of public packet-switched networks. The second segment deals with wide-area multi-drop network alternatives. This encompasses leased lines -- both analog and digital, concentration devices, satellite communication services, public packet-switched networks, all in support of an on-line business transaction system. The final segment of the case study allows the student to accomplish a preliminary local area network design, including floor plan layout, for a company's headquarters complex. The major IEEE LAN standards -- 802.1, .2, .3, .4, and .5 -- are studied and applied in the process. The emphasis throughout the course is on student problem-solving -- both on an individual and group project basis.

Two texts are used in the course, Ellis [3] for the two wide-area segments and Madron [6] for local area networks. Ellis' book appears to be unique in the network design literature. It provides a straightforward and easily understandable design methodology for wide-area networks plus a simplified yet highly useful introduction into queueing theory modelling.

The Madron text is but one of a number of adequate LAN books available. Others that meet the objectives of the course model are Chorafas [2], Fortier [4], Stallings [10], et. al.

A wide variety of supplementary readings

are used throughout the course to keep the student abreast of current developments and trends within the industry and the communications user community. These readings typically number about fifty and are selected primarily from user-oriented trade periodicals. Chief among these are: Data Communications, IEEE Communications, IEEE Network, LAN Magazine, Network World, Telecommunications, PC Magazine.

BIBLIOGRAPHY

1. Black, Uyless D. Data Communications and Distributed Networks 2nd ed., Prentice-Hall (1987).

2. Chorafas, Dimitris N. Local Area Network Reference, McGraw-Hill Book Company (1989).

3. Ellis, Robert L. Designing Data Networks, Prentice-Hall (1986).

4. Fortier, Paul J. Handbook of LAN Technology, McGraw-Hill Book Company (1989).

5. Keen, Peter G. W. Competing in Time: Using Telecommunications for Competitive Advantage, Ballanger Publishing Company (1986).

6. Madron, Thomas W. Local Area Networks, John Wiley and Sons (1988).

7. Morley, Nick. "Advanced Degrees Can Speed Career Success", Network World (January 8, 1989) pp 35-38.

8. Porter, Micheal. Competitive Advantage: Creating and Sustaining Superior Performance, The Free Press (1985).

9. Stallings, William. Business Data Communications, MacMillian Publishing Company (1990).

10. Stallings, William. Local Networks 3rd ed., MacMillian Publishing Company (1990).

11. Stamper, David. Business Data Communications 2nd ed., Benjamin/Cummings Publishing Company (1989).

12. Stamps, David. "The Tough Search for Telecom Talent", Datamation (December 1, 1987) pp 65-72.

13. Stamps, David. "Who's Teaching Telecom?", Datamation (November, 1985) pp 82-88.

14. Wiseman, Charles. Strategy and Computers: Information Systems as a Competitive Weapon, Richard D. Irwin Inc. (1985).

WRITING IN THE FOURTH GENERATION

Dean Sanders
Applied Computer Science
Illinois State University

ABSTRACT

Those of us who teach units on fourth generation languages (4GLs) face a difficult pedagogical problem. What kind of activities can we provide that will help our students progress beyond the superficial level of learning yet another syntax and help them understand the significance of these languages? The literature on writing strongly suggests that writing assignments can help the students master difficult concepts and develop the higher level skills that should be part of their education. The author has effectively used short, narrowly focused writing assignments to supplement several courses, including a course that contains major units on prototyping, fourth generation languages, and CASE tools.

THE PROBLEM

A 4GL unit is unique in that it has aspects of a programming course and aspects of a systems analysis course, but neither programming nor systems analysis is the main emphasis. Since 4GL topics are commonly offered to students who have completed one or more programming courses and one systems analysis or software engineering course, the emphasis should not be on coding details or the steps in a life cycle. Instead the emphasis should be on higher level skills such as analyzing the strengths and weaknesses of a fourth generation language and understanding how a 4GL can be incorporated into an existing environment.

The higher level skills are much more difficult to teach. The instructor must provide a classroom environment and a set of activities that will help the students develop these skills. Narrowly focused writing activities and follow-up discussions can be an effective way to help the students develop a deeper understanding of the strengths and weaknesses of fourth generation languages and of the relationship between 4GLs and other techniques for developing software. Writing activities can also help the students compare alternative techniques, help them analyze differing points of view, and help them prepare for classroom discussions.

THE VALUE OF WRITING ACTIVITIES

The literature on writing provides several indications that writing activities can help the students develop higher level skills. There are indications that the use of written language is a major factor in the development of cognitive skills such as analysis and synthesis [2]. Some writers [4, 5, 8] have argued that the process of writing is important for enhancing learning and for promoting independent thought. Others have reported successful attempts to use writing activities as instructional supplements in a data structures course and in several traditional computer science courses [3, 6].

These articles make a strong case for using writing activities to help students learn the subject matter and become independent thinkers who can discover connections between topics and can analyze new tools and techniques. I concur! I frequently teach a course

that contains major units on prototyping, fourth generation languages, and CASE tools. There was a marked improvement in the students' performance in this course after I began using writing assignments similar to those described later in this paper. Well designed writing activities definitely have a place in several courses.

DESIGNING AND USING WRITING ACTIVITIES

There are three keys to successfully using writing assignments in a class. First, the instructor and students must realize that the goal is not to produce the kind of writing that is normally associated with an English class; the goal is to produce transactional writing, writing that is intended to inform, persuade, or instruct. This is the kind of writing that employers expect from students after they graduate.

Second, both parties must realize that the purpose of these assignments is to provide a focus for learning activities that go beyond the rote memorization of facts and procedures. The students must develop an attitude that writing is a path to understanding rather than an obstacle to overcome.

Third, when preparing assignments, the instructor must realize that writing is different from speaking. A speaker has an immediate context for making and interpreting remarks, a well defined audience, and immediate feedback, sometimes nonverbal, from the audience. A writer, on the other hand, must create a context, write to an imaginary audience, and try to produce a desired response in the absence of any feedback. The instructor should provide the context, define the audience, and specify the desired effect for each writing assignment. C. W. Griffin has provided a more complete discussion of this point [5].

After years of assigning moderately long research papers, I began to experiment with microthemes as described by Bean, Drenk, and Lee [1]. They define a microtheme as "an essay so short that it can be typed on a five-by-eight inch note card". I typically expand the definition to permit one, and occasionally two, single spaced pages. Microthemes have several advantages: (1) their length makes them less intimidating to the students; (2) by using microthemes, the instructor can require more writing throughout the semester; (3) grading takes less time; (4) it requires a great deal of thought to condense the essence of a concept into a few hundred words. I have used three types of microthemes: quandary posing, thesis support, and summary writing.

A quandary posing microtheme is based on a problem whose solution is not immediately obvious. The students are required to solve the problem and explain their solution to a colleague who might have slightly less experience and knowledge.

A thesis support microtheme is based the process of evaluating alternatives rather than deriving a unique solution. The students are asked to write a statement of support for a particular position. Sometimes I let them select a position; other times I direct them to support a particular thesis.

The third type of microtheme, summary writing, is the most difficult because the students must study original materials to extract the network of ideas and understand the author's point of view. Summary writing is a particularly effective way to have the students learn to deal with controversial topics and conflicting points of view. Sometimes I use summary writing as a prelude to class discussions. Summary writing, or the related activity of preparing a topical outline, is also a good follow-up to viewing a video tape or listening to a guest speaker.

In order to establish an audience and a context, I frequently present the assignment as a request to respond to a memo, to prepare an executive summary, to write an advice column, to write a paragraph for a training manual, or to write a letter to the editor of a journal. I try to avoid asking the students to write something for a teacher to grade. Sample assignments are included in the appendix to this paper.

EVALUATING THE PAPERS

The task of grading written work can be intimidating. The key to grading these assignments efficiently is to use a simple grading scale, to grade holistically, and to minimize comments on individual papers. These suggestions make the grading easy to manage and are very appropriate for teachers who might feel uncomfortable responding to the details of the writing but who usually have a sense of whether or not a paper is well written.

I use a six-point scale for all writing assignments. This scale is wide enough to differentiate between papers but narrow enough to permit me to make decisions quickly. Papers with no significant weaknesses will receive a score of 4, 5, or 6 depending on how well the student presented the material. Papers with one or more major flaws will receive a grade of 1, 2, or 3 and will normally lead to a written comment describing the nature of the flaw. Scales similar to this have been proposed in the writing literature [1, 10].

Holistic grading is a process of responding to the writing as a whole rather than giving subscores for such things as mechanics, organization, and content. Holistic grading is effective as long as the students receive a reasonable form of feedback. One of the most effective forms of feedback is a classroom distribution and discussion of a few good papers and a few weak papers. This use of class time is not wasted because it can be used as a starting point for further discussion of the content and certainly helps the students learn to think and write in an appropriate manner.

SUMMARY

Well designed writing activities can be a valuable supplement to several courses, including those that contain units on fourth generation languages and other topics where the primary emphasis is on concepts and relationships rather than on facts and procedural details. Writing assignments help promote learning and help prepare the students to be working professionals. The benefits to the students outweigh the inconvenience of preparing and grading the assignments.

REFERENCES

1. Bean, J. C., Drenk, D., and Lee, F. D. Microtheme Strategies for Developing Cognitive Skills, in New Directions for Teaching and Learning: Teaching Writing in All Disciplines 12 (December 1982) San Francisco: Jossey-Bass, December 1982.
2. Emig, J. Writing as a Mode of Learning, College Composition and Communication 28, 2 (May 1977), 122-128.
3. Flaningam, D., and Warriner, S. Another Way to Teach Computer Science Through Writing, SIGCSE Bulletin 19,3 (Sept 1987), 15-17.
4. Fulwiler, T. Writing Across the Disciplines. Boyton/Cook, Upper Montclair, N. J., 1986.
5. Griffin, C. W. Using Writing to Teach Many Disciplines, Improving College and University Teaching 31, 3 (Summer 1983), 121-128.
6. Hartman, J. Writing to Learn and Communicate in a Data Structures Course, SIGCSE Bulletin 21, 1 (Feb 1989), 32-36.
7. Kugel, P. Improving Learning Without Improving Teaching, Computer Science Education 1, 2 (1984), 145-152.

8. Raimes, A. Writing and Learning Across the Curriculum: The Experience of a Faculty Seminar, College English 41, 7 (March 1980), 797-801.
9. Tchudi, Stephen, Teaching Writing in the Content Areas: College Level. NEA, Washington, D. C., 1986.
10. White, Edward, Teaching and Assessing Writing, Jossey-Bass, San Francisco, Ca., 1985.

APPENDIX A -- ACTIVITIES

ACTIVITY 1 (Thesis Support by an Advice Columnist)

Activity 1 is particularly difficult because the students must establish a position, must condense a large amount of material into a few sentences that support their position, and must avoid technical jargon in their replies. This assignment forces the students to wrestle with concepts that they might understand only at a superficial level. The replies also help me detect gaps in their understanding. I normally give two or three assignments of this type during the semester.

* * * * *

MIS Today is a monthly magazine written for MIS managers. This publication features both introductory and survey articles on current topics related to information systems. Technical details are not included in the articles. You are the author of the popular "Ask the Expert" column which consists of answers to questions submitted by readers. You have just received the following letter from a concerned reader.

> Dear Expert:
> I am the manager of a modest sized data processing organization. Recently, I have been reading articles about fourth generation languages because two of my employees have been urging me to purchase a 4GL. I know that a 4GL is supposed to make it easier to develop applications software, but I'm confused.
>
> We use the Yourdon/deMarco/Constantine approach to structured analysis and design for almost all of our new development, but many of the articles imply that this would be unnecessary with a fourth generation language. Can (should) structured analysis and design be used in conjunction with a 4GL? If structured analysis and design are not appropriate, how should we document the purpose and logical structure of a new software system?
>
> Just sign me: Generation Gap

Please write a response to "Generation Gap". Space limitations in the magazine require that your response be brief enough to be printed on one side of a standard sheet of paper. Naturally, these restrictions assume normal text size, normal spacing, and reasonable margins.

ACTIVITY 2 (Executive Summary for a Supervisor)

Summary writing is a particularly difficult activity because the students must separate the main ideas from examples and illustrations, must accurately represent the main ideas, and must avoid making value judgments about the viewpoints that are expressed in the articles. If a single summary is to be prepared from several different articles, the students must also identify both points of agreement and disagreement between the articles. This assignment requires the students to go beyond the preparation of a summary and add their personal interpretations. Prior to this assignment, the students should be exposed to exemplary examples of executive summaries.

* * * * *

Read the article(s) listed below (attached) then write a one or two page executive summary that synthesizes and interprets the ideas covered in the article(s). As you prepare to write the summary, you should ask yourself the following questions.

1. What theme is common to (all of) the article(s)?
2. Which aspects of this theme are covered in the article(s)?
3. If some of the articles appear to contradict one another, how can you explain the apparent contradiction?
4. Can the ideas in some of the articles be combined to form a more comprehensive point of view?
5. What is the overall significance of the articles?

Your summary should be written for your supervisor who is familiar with the general subject matter, but who has not read the articles. After reading your summary, your supervisor should know what each article contributed to the central theme, what significance you attribute to the articles, and how you have synthesized the various viewpoints. Your supervisor should be able to distinguish between the ideas expressed in the articles and your personal interpretations and opinions.

Using REXX and ISPF as an Expert Systems Teaching Tool

James A. Nelson
New Mexico State University

ABSTRACT

An existing course in online business systems design that used EXEC 2, REXX, ISPF, and CICS for traditional interactive transaction processing was to include an expert systems component. REXX and ISPF were used as the inference engine and user interface to create non-trivial expert systems, while retaining the goals of providing the students with knowledge and skills of command procedure languages and screen design tools for transaction processing. The continuing shortage of MIS faculty forces many colleges and universities to utilize all available faculty teaching hours in core courses. Schools must first teach the service MIS course and then the five or six core courses required for the MIS degree. Often, as at our university, this leaves little time or resources available to offer some of the more dynamic and interesting courses such as data communications, decision support systems, expert systems, and others. Although our Business Computer Systems (BCS) program is technically oriented toward programming and IBM mainframe computers, our BCS students have little exposure to advanced topics in expert systems and other areas for two reasons: a lack of faculty time for teaching these courses and insufficient funds to acquire production standard expert system software tools. Within a mainframe orientation, the deficit of inexpensive high quality tools is particularly acute.

The purpose of this paper is to describe how an existing required core course in our Business Computer Systems major was modified to include a strong expert systems component using software tools that are very widely available in IBM mainframe environments.

TRAINING APPLICATIONS OF INTERACTIVE VIDEO APPLICATIONS

Gerald J. Tatar
School of Business
Duquesne University

ABSTRACT

New technology continues to impact education. Better ways emerge to present both traditional and new concepts to students. One of these new technologies is interactive video. Students are able to see and review various learning situations through a self paced lesson. The military and other large entities have made significant purchases of interactive video technology in the recent past. Videodisc technology provides simulated exercises that can be difficult or dangerous to recreate. The technology has proven effective in mundane and routine technical training. Interactive video applications are becoming more useful for training employees and students from all walks of life.

A Structure for Teaching Management Information Systems

Diane M. Miller
University of Alabama

ABSTRACT

Because it is a hybrid field, Management Information Systems as academic discipline and as a profession suffers from the lack of consistent definitive description. This paper proposes a structure which employs a matrix display of MIS functions mapped against fields which are most directly involved in the planning, implementation, and operation of MIS. This structure has been an effective tool for the classroom presentation of an introduction to MIS, and it constitutes an attempt to arrive at a more definitive description of the field.

Corporate Advisory Boards: Making the Business-Education Liaison Work

Thomas A. Pollack
John C. Shepherd
Duquesne University

ABSTRACT

To the chagrin of some in higher education, more colleges and universities are collaborating with the business community on a number of issues, including the sacred issue of curriculum. One means of collaboration is the corporate advisory board, and the number of corporate advisory boards serving higher education has been increasing in recent years. Corporate advisory boards can contribute to the quality of an educational program in many ways. However, to be productive, an advisory board must be properly managed. With proper management and clearly established objectives, a corporate advisory board can contribute significantly to an institution's pursuit of educational excellence.

INTRODUCTION

In a survey and analysis done by the Association for University Business and Economic Research (AUBER) for the American Assembly of Collegiate Schools of Business (AACSB), it was reported that half of the colleges that responded at the time of the survey had some type of business advisory group and three-fourths of those without any type of group planned to start one within two years. (3) There is a decided trend for colleges and universities to seek input from the corporate environment in which they exist. Fortunately, there are many individuals in the corporate community who are devoted to civic and educational affairs and who are willing to contribute to solving problems and improving the educational process. The corporate community and higher education are always searching for a better communication pro-cess. Advisory boards provide both parties with a distinct and unique communications vehicle. The one-on-one relationships of faculty and advisory board members also enable each side to better understand the other's point of view.

The following list represents typical benefits derived from the existence of an advisory board:

- Improved contacts with employers;
- Source of new ideas;
- Improved public relations;
- Curriculum enhancement;
- Fund raising; (3)
- Corporate internship program development and enhancement;
- Career planning seminars; (1)
- Show-casing outstanding students
- Identification of local and regional business needs for continuing management development programs;
- The establishment of a natural "sounding board" for college programs in executive education, international management, and faculty internships in business;
- Match faculty expertise and resources to business problems requiring research and consultation; and (6)
- Improve the school's relationship with other schools, agencies, and professional organizations. (2)

FORMING THE ADVISORY BOARD

In general, one is likely to encounter three different types of advisory

boards: General, Program, and Ad Hoc. Each type can make vital contributions to the institution, if used properly. However, no advisory board should be formed if there is not a commitment from within the school to properly direct and administer the activities of that board.

General advisory boards can provide valuable insight and advice for the school in general. Information and advice may prove valuable in that it enables the school or university to become more sensitive to the changing needs of the business environment. This sensitivity coupled with appropriate reactions can help ensure that the academic unit maintains a "state-of-the-art" orientation. This will, in turn, help to secure community support that is often vital.

Program advisory boards are obviously formed to advise on specific programs offered by a school. Primary concerns include curricular content, equipment, facilities, and placement of graduates. Again, the pulse of the particular discipline in question is quite important, and the collaborative efforts of the academic unit and advisory group can be beneficial to both the academic institution and the business units represented.

Ad hoc advisory boards are generally short term, usually appointed to carry out a specific, frequently one-time task, or to address a special problem con- fronting the academic unit. (7) This type of board can serve a very useful purpose when input from an outside concern is important for resolution of an issue.

There is universal agreement among those who have experience with advisory boards that the most important step in establishing the advisory group is the establishment of clear objectives for that group. Objectives should be congruent with the objectives of the school and the university. They must also be realistic and attainable. If the educational unit can determine that its objectives for the advisory board can be facilitated and that a person or persons exist who will help manage the activities of the advisory board, the next step is to proceed with the establishment of a framework and charter. Once this task is complete, duties and responsibilities can be assigned to the advisory board and the management process begins.

Normal procedure calls for the election of a chairman and vice chairman of the advisory board. Other officers can be elected if deemed necessary. The chairman and the vice chairman should provide leadership and direction for the overall membership of the board. They should also work closely with the faculty coordinator of the advisory board in planning agendas, committees, and activities. This is critical to the overall success and operation of the advisory board.

In assembling the advisory board, proper care in selection of members will result in an optimal mix based on the type of organization represented, the management of the members, and the influence or community standing of the members. (3) Most institutions that have advisory boards have established rotating three-year terms for members. This allows one-third of the board to be replaced or reappointed each year once the cycle is established. Continuity is always important, and at times, turnover may also be healthy.

It is necessary to choose committed members and to require participation as a condition for membership. It is best to choose a common level of participants, but select members from diverse areas of businesses and industries. Avoid the appointment of a person who serves on an advisory board of a competing institution, especially if fund raising is involved. Survey participants in the AACSB-AUBER study identified the order of criteria for selecting advisory board members as

follows: (3)

Criterion	Rank
Major Area Employer	1
Influential Organization	2
Other	3
Veteran Business Executive	4
Civic Leader	5
Alumni	6
Major Donor	7
Fortune 500 Company	8

Appointment invitations should typically come from the dean or other appropriate high-ranking official. The dean or official should maintain an information, advisory, resource role in regard to conduct of meetings. The advisory board should be given responsibility for its meetings. Meetings should be scheduled two to four times per academic year.

The following is a summary of meeting guidelines as presented in the AACSB-AUBER Report. (3)

- Use effective, comfortable meeting room arrangements.
- Move to a clear workspace after a meal.
- Make every effort to make the members feel welcome.
- Introduce any new members or faculty guests at the beginning of the meeting.
- Begin on time and follow an agenda. Distribute a copy of the agenda at the beginning of the meeting or mail it with the meeting notice.
- Conduct the meeting so as to maintain interest and accomplish its purpose within the allotted time.
- Eliminate outside distractions.

The bulk of the work of an advisory board will generally be done in subcommittee meetings. If meetings are conducted several times per year and the length of those meetings is necessarily controlled, little time will be available for actual work activities.

The following list represents possible Corporate Advisory Board Subcommittees as presented by the AACSB-AUBER Report. (3)

- Academic Standards
- Admissions/Recruitment
- Career Advisement
- Curriculum Development
- Executive-in-Residence
- Faculty Development
- Financial Planning
- Fund Raising
- Internship Programs
- Placement
- Public Relations
- Student Aid

The above list represents compilations of responses from the deans of those schools and/or programs which have existing advisory boards. The list therefore rep-resents items designated as the result of the experiences or areas of concern of other advisory boards. The AACSB-AUBER Report is a very comprehensive summary of the survey results of those educational institutions experienced in managing the activities of an advisory board, and the recommendations contained therein are sound.

PROBLEMS TO AVOID

There are several major pitfalls which must be avoided when an advisory board is formed. The primary pitfall occurs when a college or university unit is unclear about what they want the board to accomplish. The question of what an educational unit wants from an advisory board must be thoroughly addressed before the board is formed. Effective advisory boards are the result of disciplined administrative strategy and planning. Business executives respond best to specifics. Institutions with successful campus-corporate ventures are generally the ones that have shared a list of goals with their advisory board and then give the board members an opportunity to help achieve those goals.

Another problem which must be dealt with is the difference in decision-making styles of business and higher education. Business executives are accustomed to immediate action. Higher education tends to focus attention on institutional process and therefore is typically slower to act. Decisions in higher education are sometimes bound by tradition, often at the expense of efficiency. Failure to recognize these differences can make meaningful collaboration between business and higher education difficult. (4) Awareness and willingness to mediate in the decision making process is important in ensuring smooth operation.

Like others, advisory board members like to see that their efforts are contributing to the overall success of the program and/or school. It is very important to follow-up and report (positively, negatively, or indifferently) on the recommendations made by the board. Failure on the part of the educational institution to acknowledge the accomplishment of the advisory board can result in outright failure. The exercise of common sense and good manners in recognizing the efforts of the advisory board can result in much good will from the corporate side.

THE MIS ADVISORY BOARD AT DUQUESNE UNIVERSITY

The MIS Advisory Board at Duquesne University was formed two years ago. We have adhered to most of the guidelines presented in this paper in operating our advisory board, and we are quite pleased with the outcome of our advisory board activities.

In forming the MIS Advisory Board, we sought the highest ranking information systems officer available from the major corporations in our metropolitan area. We are an urban campus and we are surrounded by a number of corporate head-quarters, so this was not a difficult task. It was our experience that nearly all who were invited were honored to serve on our MIS Advisory Board.

Organizationally, our board consists of sixteen members representing fifteen organizations. We elect a chairman and a vice chairman at our January meeting each year. We have three active committees, namely Curriculum, Marketing, and Hardware/Software. Each board member serves on one committee and each committee has a faculty liaison. Our board members serve rotating three year terms. Our Board meets three times per year, once in the middle of the Fall Semester and twice during the Spring Semester. The majority of our Board Members prefer breakfast meetings, and we typically meet from 7:30 AM to 9:00 AM. The committees meet independently at various times and various locations.

Since board members are generally results-oriented, we have chosen shorter term projects to bring before them. Included among our activities to date are the following:

- Validation of a revised curriculum for a graduate program which awards a Master of Science in Management Information Systems;
- Development of a marketing and recruitment strategy for both the undergraduate and graduate MIS programs;
- Development of a guest lecturer exchange between Duquesne University and the corporate sector represented by the Advisory Board;
- Development of student internship and employment opportunities;
- Validation of desired competencies for an entry level MIS employee;
- Deliberation as to whether the undergraduate program should continue to be called Management Information Systems or renamed Computer Information Systems; and
- Development of an MIS Scholarship program.

Agendas and background materials are

mailed to Board Members at least one week in advance of the meeting. The meetings are well-structured. They typically include announcements, committee reports, and a focused discussion topic. There are usually invited University guests such as the Director of Information Systems, the Chair of the Computer Science Department, the Academic Vice President, the Director of Career Planning and Placement, the Director of Admissions, and others. At most meetings, two MIS students are also invited and asked to briefly relate their experience with an aspect of the MIS program. The Advisory Board suggested that student input be an integral component of our meeting structure. In addition, a report of faculty research and projects underway is also submitted to the Board at each meeting. Each board member also has a copy of the master syllabus for each course offered in both the undergraduate and graduate MIS programs. The date and time of the next meeting of the MIS Advisory Board is always agreed upon by the members before adjournment.

The plan for our school is to develop a corporate advisory board for each area of concentration. The chair of each concentration advisory board will automatically be part of the overall Corporate Advisory Board for the School of Business and Administration. The chairs will be joined by other leaders of the Pittsburgh corporate community on the Corporate Advisory Board for the School of Business and Administration. The end result will be that each discipline will be linked in a specialized, collaborative way to the corporate sector. The School will benefit by having leaders from each discipline as well as other prominent corporate leaders to provide valuable insight and information.

CONCLUSION

A corporate advisory board can contribute substantially to an institution's quest for educational excellence if managed properly. Management and planning are two of the keys to success in the operation of an advisory board. On the other hand, without care, attention, and understanding, the advisory board can become a nuisance for its members and a headache for the official attempting to administer it. Good intentions, however noble, will not generate involvement without motivation and direction. (7) Board members need to be shown appreciation for their contributions, no matter how large or small. They also need to see the results of their efforts while looking ahead to future projects which will maintain their continued interest and dedication. The management style of the faculty liaison is a vital component which will help ensure communication and cooperation between the advisory board and the school.

REFERENCES

1. Cummings, Peter. "Vesting the Interest: Corporate Involvement Leads to Financial Commitment," Case Currents, November, 1981, pp. 32-35.

2. Cuninggim, Merrimon, "The Pros and Cons of Advisory Committees," Association of Governing Boards of Universities and Colleges, July 1985.

3. Gnuschke, John, Rene Elicano, and Marilyn Helms. "Business Advisory Councils: An Analysis of a Survey of AACSB Colleges of Business," A Survey Prepared by the Association for University Business and Economic Research (AUBER) for the American Assembly of Collegiate Schools of Business (AACSB), May 1985.

4. Gould, Meridith, "University-Business Partnership Must Be Well Managed To Be Productive." The Chronicle of Higher Education, June 1, 1988. pp. B1-B2.

5. Simon, Lawrence S., "The Benefits of Serving on an Accounting Advisory Board" Journal of Accounting,

January 1988, p. 24.

6. Sturgeon, C. Eugene, "The Advisory Council: It's Working for the College of Business," Illinois State University Business News and Views, Fall 85, pp. 9-10.

7. Thompson, Hugh, "Are Boards Other Than Trustees Needed?" AGB Reports, May/June 1984, pp. 27-30.

BUSINESS - ACADEMIC COOPERATION: A MUTUALLY BENEFICIAL ARRANGEMENT

Mary Martin Koleski
Computer Technology Coordinator
Purdue University - Kokomo, Indiana

ABSTRACT

Academic institutions and business/industrial organizations can cooperate in a number of ways for mutual benefit. Several are discussed in this presentation, such as individual sponsorship/ mentoring programs for selected students, Technical Preparation programs to educate future workers for the increased educational requirements of jobs, tuition reimbursement programs for employees who wish to further their educations, customized programs delivered by educational institutions in house or on campus for business/industrial organizations, and the creation and ongoing use of Business/Industrial Advisory Boards by educational institutions.

Such cooperative efforts benefit the participating organizations, as well as involved students, local communities and society as a whole.

INTRODUCTION

America's workplace is changing. Industry requires greater literacy and technical competence of its incoming workers. Current employees may not have the skills to handle the increasing rate of technological sophistication required to control and maintain new equipment. They will require upgrading in reading, mathematics and technical basics. Long-term employees need education to begin second careers to fill their estimated twenty to thirty years of life after retirement.

Business and industry are pouring unprecedented amounts of money into internal education programs. At the same time, dollars for public and private educational systems are becoming more and more scarce. Business people are not trained education experts, but they do know what skills are required by their work forces. Educators have expertise in teaching, but are not always aware of the specific, immediate needs of business and industry. Cooperation between the two sectors can yield mutual benefits.

FOCUS OF THE PAPER

This paper will discuss several programs already in existence which can promote business and academic cooperation, resulting in benefits not only to each sector but to students and to society as well. Some of the programs to be discussed are: early intervention/mentoring, "Technical Preparation", tuition reimbursement, outreach/customized curricula and the use of advisory boards by technical education institutions.

The information presented in the paper comes primarily from the experience of the author and from personal interviews. The author has worked with business and professional technical advisory boards of two educational/training programs in computer technology: a two year associate degree program which is a part of a university-based statewide technology program and is located in a medium sized Mid-Western industrial city; and a one year training program for physically impaired adults which is a part of a comprehensive rehabilitation center in a large Mid-Western city. The

author also has been involved in teaching university sponsored outreach courses in-plant for a major manufacturer as well as a local "Jobs Bank" program, and as an adviser during current development stages of a local "Tech Prep" program.

Interviews were conducted with a number of advisory board members of the educational programs, with directors of "Tech Prep" programs in four states, and with the director of a university based department which customizes educational programs for business and industry.

EARLY INTERVENTION/MENTORING PROGRAMS

Programs designed to assist disadvantaged students stay in school, earn good grades, and qualify for assistance toward college finances after high school graduation, have gained notoriety in the past few years. Many are undertaken by wealthy individuals who will "adopt" the students of a particular school or school district. Many of these programs begin with students in late elementary or early junior high school grades. For the most part, they are too new to have produced data to determine their quantitative and qualitative effectiveness. However, telecast interviews with participating students indicate that they have gained feelings of excitement about school, personal worth at being chosen for the program, hope to further their education beyond high school, and determination to continue school regardless of adverse peer pressures.

In some communities, industry has developed similar programs. Employees, particularly from minority populations, serve as mentors to students and function as counselors, boosters and role models. The companies they represent make a commitment to those students who take and pass college entrance courses to pay their expenses at a technical school or university.

Because of these programs, more eligible students are encouraged to stay in school in an era when dropping out is socially acceptable to the individual in the short term but devastating to that individual and society in the long run. Disadvantaged students are encouraged to think of long term consequences of their day-to-day actions. Special attention can increase personal gratification and help the students grow in self esteem. Disadvantaged students have a promise of being able to attend college when scholarship funds are extremely difficult to locate. Although the students are not required to make an employment commitment to the organization which sponsored them, many choose to remain and seek employment in their home towns, thereby enriching their communities and those of the sponsoring companies.

The volunteer mentors, who are employees of the sponsoring company, watch youngsters grow and develop controls over their lives. The mentors are offered a concrete way of helping themselves, benefiting their communities, and of repaying, to some extent, any help they may have received in maturing and successfully completing their own educations.

TECHNICAL PREPARATION PROGRAMS

Technical Preparation (or Tech Prep) programs are funded by matching federal and state monies. These programs, recent in development, afford linkages, primarily between high schools and technical departments in post-secondary educational institutions. They are designed to prepare general education students in high schools to take courses in mathematics, science, English/communications skills and some technical basics courses designed to teach these subjects in a new way. The courses are to be taught in a "hands-on", inductive approach to students who do not thrive in a theoretical, deductive mode of education. The legally specified cooperation is between high schools and post-secondary

technical schools which design the curricula jointly. The most successful "Tech Prep" programs around the country involve not only the schools, but local industries as well. Each partner makes a significant contribution. [10]

INDUSTRY CAN CONTRIBUTE IN THE FOLLOWING WAYS:

(1) Make known the current and anticipated requirements for future workers, many of whom will be products of local schools.
(2) Offer summer internships to Tech Prep faculty to allow them to learn real world applications within their subject areas to supplement textbook learning.
(3) Host field trips and lectures for Tech Prep students to make them aware of areas of need for future employees and the day-to-day work expectations of employers.
(4) Make equipment (CAD systems, for example) available at the company site to Tech Prep classes in off-hours. This may require modifications in the traditional attendance hours of high school students, but may offer course work which insures future employees who have the skills required to be productive immediately upon being hired by industry. [8]

POST-SECONDARY TECHNICAL SCHOOLS AND COLLEGES AND UNIVERSITIES WITH TECHNICAL PROGRAMS MAKE THE FOLLOWING CONTRIBUTIONS:

(1) Assist in planning the new curricula and aid in its implementation.
(2) Arrange the articulation of the "new" courses with existing entrance requirements.
(3) Offer advanced standing in their technical programs for students who have successfully completed Tech Prep credits. [3]

HIGH SCHOOLS, WHICH RECEIVE 70% OF THE FUNDING, MUST AFFORD:

(1) The location for the program, either for their students alone, or for the students from a regional cluster of schools.
(2) Teachers who are willing to change from a primarily deductive to an inductive mode of teaching.
(3) Time for these teachers to prepare new interpretations, applications and approaches to traditional subject matter.
(4) Retraining of staff, such as guidance counselors, to value the programs and help to devise tests for appropriate candidates and to recruit eligible students. [7]

EMPLOYEE TUITION PROGRAMS

TUITION REIMBURSEMENT

Employee tuition programs have two features: the types of courses for which employers will pay; and the mode of reimbursement used by the company. First, some companies will pay tuition for any college or university course taken by employees; other companies reimburse their employees only for those courses or programs which are job related or job enhancing. As to the amount of tuition reimbursed, some companies pay full tuition for courses; others reimburse their employees on a sliding scale, depending upon the grade earned in the course (for example, full reimbursement for an earned "A", half tuition for a "B" and nothing for a "C" or less).

THE "JOBS BANK" PROGRAM

Another example of industrial-educational cooperation has been developing in the past few years. In this, labor unions play a significant role as well. One example is a program co-sponsored by a major national employer and a major labor union. Long-term employees, who may be displaced by anticipated relocation of local assembly

lines, are sent to technical programs of post-secondary schools full-time to be retrained. Some of these employees will move into jobs remaining in town, while others will be capable of going on to jobs outside the manufacturing field. Eligibility for the program is determined by seniority. The workers, who receive their regular salaries and benefits during the course of this "Jobs Bank" program, are replaced in the factory by workers with less seniority who previously had been laid off.

CUSTOMIZED EDUCATIONAL PROGRAMS

Customized educational programs fall into two categories: teaching existing college or university courses in-house for company employees, or designing specific courses for groups of employees in a company.

Taking existing courses to a company site is particularly helpful to shift employees whose work schedules will not always allow them to attend post-secondary schools at "normal" school hours. Many courses can be easily transported by the instructor. However, laboratory classes to not lend themselves to this arrangement, and the students usually must come to the campus for these courses. The class hours, examples and assignments for such courses can be made relevant to the students' employment situation.

Some colleges and universities have departments which will work with companies to design courses, either credit or non-credit, to the company's specifications. These may even include visual aids showing company sites, designed to train workers on particular company machines or to follow a company's unique procedures. [11]

ADVISORY BOARDS

Advisory Boards have been used by many educational/training institutions to assist in a number of ways which are of mutual benefit to themselves and to the business industrial organizations which participate. Such boards are frequently used to form boundary spanning linkages between business/ industrial organizations and academia, particularly by technical departments, to insure high quality, up-to-date programs which will meet the educational needs of the students and the occupational needs of the business community. The advisory board members serve as volunteers and are generally chosen for a particular type of expertise they possess or for their influence in the community. The board facilitates communication, understanding and cooperation between those who are responsible for an educational program and those in the community to whom that program is relevant. [5]

Some of the activities discussed under the "Technical Preparation" programs are undertaken by advisory boards, but are not limited to those already mentioned. The educational institutions gain some of the following benefits through the use of advisory boards:

(1) Advice from experts on the acquisition or updating of technical hardware and software.
(2) Assistance in obtaining equipment for laboratories through grants from the industry, or through donations of equipment that is being replaced by the industry but is newer than that available in the school.
(3) Reviews of curricula to insure that the school is producing graduates with the state-of-the-art skills required by business and industry.
(4) Guest speakers in classes to inform students of current practices and of career possibilities in the "real world".
(5) Sites for field trips for students to supplement classroom instruction.
(6) Sources of internship and cooperative education programs for students.

(7) Qualified adjunct faculty to supplement full-time faculty.
(8) Placement opportunities for program graduates.
(9) Opportunities for summer employment of faculty to enable them to earn income year-round and to stay current in their fields.

Colleges and universities cannot offer tangible rewards to participating companies or advisory board members, but indications are that participants on such boards gain many intangible rewards which are key factors leading to energetic and effective service. Some that have been identified are:

(1) Contributing to a program that helps to shape and develop students' abilities and future prospects.
(2) Assuring the companies and/or the participants own departments of a pool of competent potential employees.
(3) Engaging in intellectual problem-solving in a new and different setting.
(4) Acquiring status and prestige at home and on the job by being associated with a prominent and respected community institution.
(5) Gaining new skills in handling interpersonal situations.
(6) Increasing opportunities for supervisory trainees to gain managerial experience through serving on a community project.
(7) Gaining the sense of being of service to others [2, 5]

CONCLUSIONS

There are a number of ways in which businesses and academia can cooperate. At the same time, they must adjust to each others mores. Business must learn to be tolerant of the slow pace at which academic institutions are able to change directions, as when modifying curriculum. They must be open to the budget limitations under which schools must operate. Educational institutions, on the other hand, must be willing to accept the fact that outside "amateurs" have a stake in the products of schools, and are experts in the skills they expect find in newly hired employees.

Whether business and industrial organizations participate on advisory boards, or serve as partners in education through tuition reimbursement, "Jobs Bank" or "Technical Preparation" programs, or arrange for customized educational programs or individualized mentoring programs with specific students, their influence is enormously important to the success of modern educational programs.

REFERENCES

1. Blackstone, Bruce I. "Say "Yes" When You Are Asked to Serve." Office. (1974) 79(1). 112-114.

2. Blanks, Edwin E. "Cracking Classroom Isolation: Education's "Working" Partnership." Data Management. (1977) 15(1). 29-31.

3. Hata, David M. Curriculum Coordinator, 2 + 2 + 2 Tech Prep Program, Portland (OR) Community College. Personal interview, 7 December 1989.

4. Heinje, Cynthia L. "Faculty Develops Curriculum With Guidance of Business." (1975) Data Management. 13(3). 18-19.

5. Koleski, Mary Martin, Ho, Tom I.M. and Koleski, Raymond A. "Building the Bridge Between Business and Education: Using Advisory Boards Effectively. "Proceedings, International Business Schools Computer Users Groups Conference.(1987)

6. "Making Committees Work." Infosystems. (1985) 32(10) 38-39.

7. Marmaras, Judy. Project Director, 2 + 2 Tech Prep Associate Degree Program, Community College of Rhode Island. Personal interview, 4 January 1990.

8. Powers, Kim. Project Director for Tech Prep, Office of Vocational Education, Indiana State Department of Education. Personal interview, 7 February 1990.

9. Status Report on the Development of the Technology Preparation Curriculum Models, Submitted to the Indiana General Assembly by the Technology Preparation Task Force, November 1, 1989.

10. Tilmans, Anthony L, Ph. D. President, Kansas College of Technology. Personal interview, 3 December 1989.

11. Wiersteiner, S.R., Executive Director, Vocational-Technical Center, Indiana State University, Personal interview, 7 February 1990.

CORPORATE-UNIVERSITY EDUCATION PROGRAMS: FORGING A RELATIONSHIP OF TRUST

Stuart A. Varden
Frank J. LoSacco
Information Systems Department
Pace University

ABSTRACT

This paper discusses eight areas in which corporations and universities differ in their approach to developing, administering, delivering and evaluating educational programs. The paper suggests an approach to resolving each of these differences so as to facilitate the collaborative creation of corporate-university education programs that will serve the interests of both parties. The authors draw heavily from their experiences with the Graduate Telecommunications Program, a joint education program offered by IBM Corporation and Pace University.

INTRODUCTION

Over the past two and a half years, IBM and Pace University have been involved in a joint program called the Graduate Telecommunications Program (GTP). That this ongoing program has worked out successfully for both participants, and continues to do so today, is due largely to the development of an effective working relationship between IBM and Pace.

This paper addresses this relationship. It contrasts the approach to higher education as viewed by a major corporation and a major university. It then describes what the two organizations did to bring their representive points of view together to permit them to accomplish a program that has met both their sets of goals and expectations.

BACKGROUND

In the fall of 1987, IBM approached the university community with an invitation to work jointly on an intensive credit-bearing graduate level program in telecommunications. After several interactions, Pace University was selected to implement a program in which both IBM and the university were interested.

Since early in 1988, the authors have been involved in developing, offering and evaluating this IBM Corporation sponsored graduate-level education program in telecommunications, called the Graduate Telecommunications Program (GTP). We were members of a small curriculum development team that helped develop an integrated, credit-bearing sequence of six courses designed to provide IBM internal telecommunications professionals with a solid academic background in telecommunications technology and management. The effort was an active partnership between IBM and Pace University, which has resulted in a highly successful corporate program. Moreover, the six course sequence is now the core of Pace's Master of Science in Telecommunications degree program, which is offered through Pace's School of Computer Science and Information Systems.

The Graduate Telecommunications Program emphasizes the role of telecommunications in achieving the operational and strategic goals of organizations. A prominent illustration of this is the recent trend toward

physical and logical network consolidation to improve the cost effectiveness of the telecommunications function.

The program is held at IBM's International Finance, Planning and Administration School, which is located on a Pace University campus in Briarcliff Manor, New York. Thus, GTP students are away from their normal work locations and understand that they are in an academic environment that encourages the open exchange of information and ideas. The views of all students are treated equally, regardless of what rank they may hold back on the job.

The six courses that make up the program are:

- Data Communications
- Telecommunications Management
- Digital Telephony and Switching
- Computer Networking
- Telecommunications Policy and Environment
- Cases in Telecommunications

A description of the program, its development, curriculum content, administrative procedures and benefits to the corporate sponsor can be found in several places (see [1], [2], [3], [4], [5]). Based on our experiences with the GTP program, we now comment on eight areas where we see significant differences in attitudes, perspectives, traditions and practices between corporate and university education. We then offer a suggested approach to resolving these differences so that both corporate and university interests will be served and a productive partnership can develop.

It should be emphasized that the differences reviewed here are drawn from our experiences and may not apply to all corporate-university education programs. Nevertheless, we feel that the differences we encountered are typical of those that most universities will find when developing a joint education program with a corporation.

AREA OF CONTRAST NO. 1: PURPOSE OF EDUCATION

THE CORPORATION

Corporations want to see a direct connection between what is being taught in the classroom and its application to the job. Usually the purpose of corporate education is to train employees in specific job-related skills that will result in an immediate improvement in employee performance and productivity. Some companies also take a more long term view; they understand that education is strategic to fostering organizational change and career development. Whether the objectives are short or long term, education is generally sponsored only when their is a clear business need. Thus, there is little interest in education that concerns itself with knowledge for its own sake or the study of a theory having no business-relevant applications.

THE UNIVERSITY

University education is more holistic in its perspective. Although the teaching of skills that prepare students to enter a profession is an important mission of any university, the tradition of intellectual inquiry leading to the advancement of knowledge with no necessary practical application in mind is greatly valued. Moreover, the goals of fostering responsible citizenship, enhancing one's quality of life through an appreciation of the sciences and humanities, and encouraging spiritual awareness, are all integral parts of university life.

SUGGESTED APPROACH FOR RESOLUTION

The differences in the purpose of education just cited are perhaps the leading reasons why corporate-university educational programs never materialize. It is clear that corporations will not

sponsor an education program that is not business relevant, and it makes no sense for university representatives to argue against this position. It is appropriate and reasonable, however, to question what is meant by "business relevant." The debate that is most likely to occur deals with the "training versus education" issue. Corporations are usually interested in narrowly defined instruction (training) that is highly task or situation specific, whereas universities generally stress a broad and generic treatment of a body of knowledge (education).

If the corporation is not seeking university credit for the instruction it is asking the university to provide, the university must simply decide whether it wishes to be in the business of offering non-credit continuing education for a fee. If, however, the corporation wishes its employees to earn credit while receiving business relevant instruction through the university, the university must make it clear to the corporation that narrowly defined, task specific training is usually not eligible for credit.

The approach we suggest is to point out that in the long run, broadly defined education is more practical than narrowly defined training. Narrowly defined training can quickly become out-of-date, while an understanding of basic concepts remains valuable for a long time, perhaps indefinitely. A middle ground position is to propose to develop a program that covers the heart of the accepted university curriculum in the desired subject, but illustrates broad concepts with the specific skills that the corporation wants its employees to learn. Thus, the skills are placed in a context that will make them more meaningful, and will help facilitate the learning of updated skills at a later time.

An example might be the corporation that wants to teach a group of its employees a database management product such as SQL/DS. A course that teaches only a specific product is generally not considered "credit worthy." However, a course in database management systems that uses SQL/DS as a vehicle for demonstrating relational database concepts would be acceptable for credit, and in the long run, of more value to the employee and corporation.

AREA OF CONTRAST NO. 2: THE AUDIENCE AND ITS BACKGROUND

THE CORPORATION

The audience targeted for educational offerings in a corporate environment is dramatically different from that found in an academic setting. The corporate student operates within the context of a specific job that he or she either has or is preparing to undertake. Further, the student customarily has some level of training or experience in the subject matter of the course being taken, or has worked closely with those who do.

The class itself, in corporate education, is comprised of a body of students which is, at least to some extent, monolithic in that all its members work for the some organization. Thus the goals, objectives, policies and procedures of the organization are the same for all the students in the class. Similarly, the corporate strategies, plans and applications relative to the technology being taught are uniform and generally understood by the class participants.

Members of the corporate class often know each other as former or current co-workers within the company. Further, class members may be related organizationally. Students in some of our GTP classes have included managers as well as their subordinates. The backgrounds of members of corporate classes usually reflects a spectrum of educational levels. It may well be that one finds high school graduates, bachelor's level students, master's

level participants and, in one case, a Ph. D. level student, all in the same class.

THE UNIVERSITY

The corporate characteristics given above are not generally found in the university classroom. Here students often do not have an actual job or any related job experience that they can bring to the classroom. Even when students do work, they are not all employed by the same company.

Thus, classes are never made up of students all with the same understanding of why and how the subject matter under discussion is to be considered, planned for or applied.

Finally, members of a given class in a university have been screened according to academic standards that ensures that the educational level of each student is roughly the same, be it undergraduate or graduate.

SUGGESTED APPROACH FOR RESOLUTION

Many of the distinctions between the two audiences and sets of backgrounds noted above actually turn out to be positive factors in terms of teaching graduate level courses in the corporate environment. That the students are, in general, more experienced in areas relating to the subject being presented actually enriches the presentations and resulting discussions.

With respect to the notion of all students coming from the same company, our corporate partner and we recognized that this could produce restricted interpretations of the course material. Thus, throughout the course development cycle, as well as in the delivery of the GTP courses, we both insisted on and continue to stress, a comprehensive coverage of the material without unwarranted corporate accents.

The most difficult aspect of the corporate-university diversity with respect to students is the mixture of backgrounds in the corporate class. This has been particularly evident to us when it comes to the student's background knowledge of mathematics and basic physics. Many corporate students have not dealt with such subjects since their own earlier student days which, in a great number of cases, might have taken place 10 or 20 years ago, sometimes even more.

IBM and Pace have worked out what appears to be a rather successful means of overcoming this difficulty -- a remedial mathematics and basic electricity course for students planning to become members of the GTP program. This course is given to students who do not attain an acceptable score on a GTP pre-test or who do but still desire a refresher course. It is four full days in length and includes both lectures and workshop sessions; the first three days are devoted to mathematics, the last day to electricity.

AREA OF CONTRAST NO. 3: STUDENT EXPECTATION

THE CORPORATION

The corporate student enters the classroom believing that the purpose of his or her being there is, in fact, to be trained. More than likely broad concepts, trends, theoretical foundations, historical considerations and the like are not of particular interest to this class member. The need, it is assumed, is to be trained to perform some specific set of tasks in the job environment.

Reinforcing this is the fact that corporate courses rarely involve university credit. Hence, student grades are usually not an issue. Once again, learning is driven by what the student interprets to be the immediate applicability, in the job setting, of the material being presented.

THE UNIVERSITY

In contrast to the above, university students do not expect to take courses that relate specifically and necessarily to the details of a specific job. Pace does not offer a credit course whose purpose is to teach how to use a specific piece of hardware or software, for example.

This is not to say that students in the university do not expect to find anything of an applicable nature in the classroom. Clearly they do. The presence of academic computer centers at universities, where students may go to use mainframes as well as personal computers, and all the software associated with these devices, is evidence of this.

However, students expect that presentations of such technical applications are, in fact, part of a set of broader, integrated conceptual considerations.

SUGGESTED APPROACH FOR RESOLUTION

These two approaches to education were clearly and thoroughly discussed by IBM and Pace in the process of developing the courses of GTP. The benefits of student exposure to background material, the theoretical bases supporting practical concepts and the need for understanding general principles were delineated. The desirability of the balance of generic and practical material usually found in the university classroom was understood clearly by both IBM and Pace and the potential problem avoided.

AREA OF CONTRAST NO. 4: RESOURCES AND COMMITMENT

THE CORPORATION

When a corporation decides that it is in its best interest to pursue a particular course of action, it allocates the needed resources, develops a plan, and puts the plan into action. The resources include money, people, space, equipment and whatever else is considered necessary to achieve the objective. The allocation of resources, then, is a clear expression of the corporation's commitment to the project. This expressed commitment to get the job done helps to ensure the success of the undertaking.

THE UNIVERSITY

Universities usually do not have a ready source of funds and staff to devote to the development of new program initiatives. Instead, initiatives begin modestly with little financial backing. Substantial institutional backing generally only comes after it becomes clear that sufficient groundwork and progress have been made to justify the backing. We might summarize this as follows: In corporations, funds tend to drive a new initiative, while in universities, funds tend to follow a new initiative that is already underway.

SUGGESTED APPROACH TO RESOLUTION

Corporations expect to pay for services rendered, and this includes educational services. This of course is wonderful for the university, but strange as it may seem, university faculty often have to get used to the idea. For example, corporations commonly refer to their outside providers of educational services as "contractors" or "suppliers" -- terms quite alien to the university world when applied to education.

In the case of the GTP Program, IBM allocated substantial funds for curriculum development. This included the full time services of four professionals (three from IBM and one Visiting Professor from Pace) for nine months, the part time services of about ten other faculty, secretarial support, office space, computer hardware and software, texts, graphics services, travel expenses, lodging, and whatever else might reasonably contribute to the

success of the program. And these expenses, well in excess of $300,000, did not include the cost of classroom instruction.

It is clear that IBM or any other corporation would not have made this level of commitment to a risky and high profile undertaking if it was not confident that it was working with a trusted university partner, a partner that would live up to its end of the bargain.

What does this mean in practical terms for the university? It means that the university must pay attention to the little things that will create a high comfort level within the corporate management responsible for the project. It means showing up to meetings on time. It means keeping to agreed upon schedules. It means dressing in accordance with company standards when visiting the corporation. And above all, it means keeping corporate management informed of the progress of the project every step of the way.

AREA OF CONTRAST NO. 5: MANAGEMENT PLANNING AND CONTROL

THE CORPORATION

The corporation's approach to managing the development, delivery, and evaluation of an education program is no different from its approach to managing other corporate projects. There is a clear management chain of responsibility, a project plan with clearly stated objectives and assignments, a schedule of key project "milestones," and periodic status reports and briefings to keep management appraised of the project's progress. If the project is falling behind schedule or going over budget, the project will receive increased management attention.

THE UNIVERSITY

A university with its strong tradition of academic freedom embraces a wide variety of working methods and individual styles. Faculty members like to do things "their way" and the freedom given to the individual is one of the major attractions of university life. This less structured approach is likely to lead the corporate partner to feel uneasy.

SUGGESTED APPROACH TO RESOLUTION

It has been our experience that the best way to gain the confidence of a corporate partner is to maintain good communications and to adhere to the management expectations of the corporation. This means plenty of meetings, project progress reports, and presentation of tangible work products. This is particularly true during the early stages when the corporation may still be uncertain about the ultimate success of the project. If it is not clear to corporate management that the project is proceeding smoothly, the natural reaction will be to take control of the process. This can lead to a kind of micro-management that interferes with the development team's ability to continue working on the project.

In the case of the GTP program, several days during the early stages of the project were devoted to preparing status reports, presenting the reports to management, and interpreting management's response. From the perspective of the development team, this was an inefficient use of time, since no work on the project itself could take place on these days. Once management was convinced that the project was "on track," the call for status reports was less frequent.

We do not wish to give the impression that it was only the university that adjusted its working style to conform to corporate expectations; the corporation also adjusted to the university. IBM management grew to realize that too much interference would be counter productive in the long run. They learned to accept the less structured management style of

the university as long as there continued to be evidence of tangible progress. This understanding of differences in working style continues to this day.

AREA OF CONTRAST NO. 6: EDUCATIONAL SETTING AND SCHEDULING

THE CORPORATION

Most corporate education takes place in a dedicated education facility. The classrooms are often technically more sophisticated than university classrooms. Two overhead projectors, white boards, video projectors for computer demonstrations, comfortable chairs and long tables, and wall-to-wall carpeting are standard. A "class manager" provides administrative and logistical support. Class sessions are held throughout the work day with student homework and study group activities taking place in the evening. The pace is very intense, and a course can be completed in a single week. It has been the practice in the GTP program to run two courses concurrently -- one in the morning and the another in the afternoon -- over a two week period. This allows students a little more time to reflect on the concepts and issues of a course.

THE UNIVERSITY

The physical classroom setting at a university is usually quite modest when compared to corporate educational facilities. Although audio-visual support is usually available upon request, it is not a standard feature of the instructional setting. The professor's class notes and a piece of chalk still prevail.

Class sessions are usually held from one to three times per week for an hour or two over a period of three to four months.

SUGGESTED APPROACH FOR RESOLUTION

Many university faculty feel that the intensive "fire hose" approach to education often found in the corporate world is educationally unsound. The feeling is that students need a longer period of time for the material to "sink in."

Although it is true that individual term papers or extensive projects are not feasible in this compressed schedule, we can report that GTP students perform as well or better on examinations compared with students taking equivalent courses offered at the university. This favorable result may be due in part to the high caliber and motivation of the students who attend the GTP program. We also believe that the superior classroom facilities and support leads to a more productive educational experience. Although the education setting is different, we feel that academic standards are of a high quality.

AREA OF CONTRAST NO. 7: MANNER OF PRESENTATION

THE CORPORATION

As with most corporate education, overhead foils play a significant role in presenting the course material. In the GTP program it was IBM's thought that all material be presented via the foil method, with textbooks used for out-of-classroom assignments. Bound copies of all foils used in a course were to be made available to students at the outset, thus minimizing the need for extensive note taking.

THE UNIVERSITY

The delivery of a university course generally takes place in a classroom whose resources are limited. Most often a professor has his or her notes and a blackboard or whiteboard.

In this environment the professor has leeway to modify the material being

presented as he or she wishes, as the students have no previously distributed set of notes to follow.

SUGGESTED APPROACH FOR RESOLUTION

University faculty were requested to develop foils needed to deliver the course in the manner expected by our corporate sponsor. This request was opposed by one professor, who voluntarily withdrew from the program. All other professors accepted the challenge of organizing their course materials in this manner. Recent discussions with these faculty members indicate that they have found the exercise to be an extremely useful one and claim that it has benefited them in their regular university course work as well.

AREA OF CONTRAST NO. 8: EVALUATION PROCESS

Most corporate education programs take place during working hours. This is quite costly, since the corporation pays all tuitions, fees, texts and material expenses, employee transportation, lodging, meals, and may even maintain an education facility where classes are held. Moreover, while students are in class, they are absent from their normal job responsibilities. Thus, it is understandable that the evaluation procedures for education programs used in corporations differ considerably from those found in traditional university instruction.

Summarized in Table 1, below, are the most significant differences in the approach to educational evaluation that we have found.

SUGGESTED APPROACH FOR RESOLUTION

Understandably, the corporate sponsor is interested in verifying that the value received in its educational programs is "worth" the cost. As a result, the evaluation component of education is taken very seriously. Interestingly, the focus of the evaluation efforts in corporate education has tended to be on the quality of instruction and curricula, while universities have traditionally focused on evaluating student performance through examinations.

These differences mean that the typical university instructor may well have to make significant changes in his or her instructional approach to be successful in a corporate educational setting. Although it can be uncomfortable or even intimidating for university faculty members to be observed and evaluated on a frequent and ongoing basis, we are convinced that instructors who are able to be self-critical will undoubtedly improve their classroom effectiveness while maintaining university academic standards. It is our observation that university instructors who have taught in a corporate setting feel that they are now better organized, are more conscious of the need to develop instructional objectives, and devote more time to preparing instructional materials for their university courses.

CONCLUDING COMMENTARY

The authors have reached a number of conclusions based on their experiences with the GTP program. First among these is that very successful corporate-university working relationships, where both participants benefit, are certainly possible and indeed desirable.

Next, it is clear that an integral part of the process of working well together involves the recognition, by both parties, that differences in their approach to education exist. This laid the groundwork for us to sit together to specify what these differences are and to constructively and mutually evolve effective solutions to any problems they might bring about.

Finally, the authors feel that during

Table 1

UNIVERSITY INSTRUCTION	CORPORATE INSTRUCTION
Course/Instructor Evaluation:	
Students fill out a course evaluation form at end of the semester.	Students fill out evaluations of each instructional unit throughout the course. Two such evaluations per day is typical. In addition, overall course evaluations are completed.
Classroom observation by administrators and other faculty occurs rarely if at all.	Class manager, other corporate staff or faculty present most of the time.
Student evaluations are one factor among many used in making instructor retention decision.	Student evaluations have direct bearing on instructor retention.
Student Evaluation:	
Graded assignments, quizzes, term paper, mid-term and final examination.	Either upgraded, or a single exit test to measure degree of learning.
Post-Program Evaluation:	
None.	Some corporations, such as IBM, study whether or not material learned in class is being applied to the job after the employee has returned to the workplace. The results of such studies can lead to changes in the curriculum or teaching methods.

the process of working together, a mutual trust in each other developed which ultimately permitted us to go on and successfully complete the work undertaken in GTP.

The authors feel quite positively about the ability of universities and corporations to work well together, and we encourage the active exploration of such relationships by other members of both communities.

REFERENCES

[1] Dines, Robert S. "IBM's Graduate Telecommunications Program," Communique, Vol.42, No. 1, February, 1989, pp. 23-25.

[2] LoSacco, Frank J. and Varden, Stuart A. "An Integrative Telecommunications Case Study: Pulling the Piece Together," Proceedings of Annual IBSCUG Conference, 1989, pp. 356-361.

[3]Varden, Stuart A. "A New Graduate Telecommunications Program: The University and Corporate Worlds Join Hands," Proceedings of the National Educational Computing Conference, 1989, pp. 65-70.

[4] Varden, Stuart A. and LoSacco, Frank J. "Facilitating Intracorporate Cooperation: A University Creates the Environment," ACM SIGCSE Bulletin, Vol. 22, No. 1, February, 1990, pp. 152-156.

[5] LoSacco, Frank J. and Varden, Stuart A. "Corporate-University Education Programs: How the University Can Benefit," Proceedings of Annual IBSCUG Conference, 1990.

ALTERING FRESHMAN VIEWS OF INFORMATION SYSTEMS CAREERS

Charles H. Mawhinney
David R. Callaghan
Bentley College

Edward G. Cale, Jr.
Babson College

ABSTRACT

College student interest in computer-related careers has declined dramatically in recent years. This trend is beginning to have a profound effect on both academia and industry. One possible explanation for this decline is that students hold negative perceptions of the workstyle associated with the positions held by information systems (IS) graduates. A study of freshman business majors was conducted which examined whether an introductory computing course changed those perceptions. The study indicated that perceptions could be changed and subsequent enrollment patterns in IS courses altered through a concerted program to change student attitudes through the introductory computer course. The results confirm that internal recruiting strategies can be successful and suggest the need to consider some of the related ethical issues. [STUDENT ATTITUDES, STUDENT PERCEPTIONS, CURRICULUM, WORKSTYLE, ENTRY-LEVEL POSITIONS, JOB PREFERENCES, INFORMATION SYSTEMS CAREERS.]

INTRODUCTION

Student interest in computer-related careers has risen and declined like a roller coaster in the last decade. In the annual survey of college freshmen conducted by UCLA (1, 2, 3, 4, 5), interest in programmer/analyst occupations surged from 2.8% in 1977 to a high of 8.8% in 1982. This unprecedented rise in interest was then followed by an equally dramatic decline in interest to a low of 1.9% in 1989.

Colleges and universities, many of which struggled to add computer-related curriculum capacity during the early and mid-1980s have already felt the sting from declining enrollments, and are now trying to maintain the viability of their programs. Industry, on the other hand, is just now feeling the effects of the reduced number of graduates.

At Bentley College we have experienced this phenomenon directly and have watched enrollment in the Computer Information Systems (CIS) major drop from a high of 681 in 1983-84 to a low of 100 in 1989-90. We have been concerned because while the supply is dropping, we do not see any decline in job opportunities for our graduates. If starting salaries are an indication, CIS majors at Bentley are in great demand, with median starting salaries substantially higher than any other major.

This decline in interest has not gone unnoticed and has been the basis for articles in the Wall St. Journal (8) and Computerworld (13). While a number of hypotheses are espoused in these articles and others (11), little organized research has been done as to the cause of the decline.

We think that a major reason for this decline in interest and enrollments may be due to incorrect perceptions held by graduating high school students. We

suspect that two important causes of these misconceptions are: (1) turmoil in the hardware industry resulting in widely publicized layoffs, and (2) students being initiated to computing through programming courses. We believe this results in freshmen who enter college with an incorrect belief that a career in computer related fields is unstable and primarily involves computer programming, and thus avoid such majors and occupations because of these misconceptions.

THE FIRST EXPERIMENT

Accordingly, we began a study of freshmen business majors in September 1987 in which we attempted to both measure and change their perceptions of the entry level position of the CIS major. During the first year (1987-88) we measured perceptions held by the freshmen and the CIS faculty following the schedule indicated in Table 1. Various comparisons were made which have already been reported (6, 9, 10).

TABLE 1. THE FIRST EXPERIMENT

DATE	GROUP	PERCEPTION MEASURED
9/87	Freshmen	CIS Grad's Workstyle
9/87	CIS Faculty	CIS Grad's Workstyle
4/88	Freshmen	CIS Grad's Workstyle
4/88	Freshmen	Own Job Workstyle

HYPOTHESIS

This paper reports on a portion of the second year (1988-89) of the study and focuses on changes that occurred in the students' perceptions of the CIS graduate's workstyle during the academic year. We believe the changes in perceptions that occurred were primarily the result of changes in the introductory computer course which were made as a result of the experiment in the prior year (1987-88).

Stated in null form, the hypothesis tested was:

Ho: Students' perceptions of the workstyle of the typical CIS graduate will not change as a result of an introductory computing course.

During the fall semester of the 1988-89 academic year, all freshmen students were enrolled in an introductory computing course. The course focused on hardware concepts, software concepts, and personal computing. School policy required all students in this course to lease or purchase an MS DOS compatible portable computer. The course was conducted using a lecture and laboratory methodology. "Hands on" applications used packaged software only (spreadsheet and word processing). No procedural programming languages were used or taught in the course. All sections had a common final exam and "hands-on" lab final, and were taught from a common syllabus which included a session on "careers in CIS". During the previous year (1987-88), this session on CIS careers included a presentation by two "guest" faculty members which generally followed a common outline developed by an ad hoc committee of senior faculty members. We anticipated this introductory course would result in a change in the students' perceptions to be more like those of the CIS faculty. However, in 1987-88 we found the introductory course had a negligible effect on changing the students' perceptions, and, if anything, the effect of the introductory course was to convince the students that the CIS graduate entered the workforce as a programmer (9). When we compared the students' perceptions of the CIS graduate's workstyle with their perceptions of the jobs they would have after graduation, we found: (1) they would prefer a job with more human interaction and less direct involvement in the implementation of computer

technology, (2) they had unrealistic expectations about their own starting salaries, and (3) they did not perceive themselves to be different from CIS majors with respect to math ability or prior computer background (10).

As a result of this experience, we modified both the content and delivery of the CIS careers session for the Fall 1988 rendition of the course. A package of instructional materials was developed which emphasized: (1) the starting salary advantage of CIS majors relative to other business majors, and (2) the diversity of career paths available in the profession. The career path aspect strongly indicated that most of the jobs were as systems analysts in user companies rather than programmers in vendor companies. As a consequence of this session in the course, we expected the students' views to shift in a direction more favorable to the CIS major. These anticipated shifts in means of the students' scores were assessed through t-tests for paired comparisons.

METHOD

Subjects

This study used as its subjects all students enrolled in an introductory computing course at Bentley College in Waltham, Massachusetts. Bentley is a private school and is the eighth largest undergraduate college of business in the United States. Almost all the participants were first semester freshmen business majors. Participation in the study was voluntary and participants were allowed to remain anonymous. However, participants were encouraged to identify themselves in order to facilitate an eventual "post treatment" survey. For the first survey (PRE), responses were received from 491 of the 1058 students, resulting in a response rate of 46%. For the second survey (POST), usable responses were received from 210 of the 491 students who responded to the first survey.

The Instrument and Procedure

A nine-item questionnaire was developed in the previous academic year to assess perceptions of the workstyles of graduates of the CIS program (see Exhibit A). The items covered a range of activities and aspects that seventeen and eighteen year olds with no prior business or computer background could reasonably be expected to understand. The first two items dealt with perceptions of general technical background. We felt that if students believed that CIS majors had a relatively stronger math and/or computer background it could be a barrier to entering the major. Item #8 assessed their perception of the starting salary of the CIS graduate. We felt this to be an important issue in a business curriculum which presumably advocates the profit motive. Item #6 was included to determine whether these students could distinguish normal CIS activities from the more technical activities engaged in by engineers and computer scientists. The five remaining items (#3, #4, #5, #7, and #9) focused on aspects of workstyle which generally distinguish computer programmers from systems analysts (7, 12, 14). The responses to the questionnaire were scored "after the fact" by assigning numeric values on a scale of 0 to 4 (SD = 0, D = 1, U = 2, A = 3, SA = 4).

The instrument was administered two times in this study. The first time it was administered to the student group, this instrument was contained within a larger questionnaire that also assessed: (1) previous computer background, (2) computer anxiety, and (3) Jungian personality type. The questionnaire was distributed in class during the first week of the course (in the Fall semester). The questionnaire was completed outside of class and returned either through the instructor or campus mail.

EXHIBIT A. THE INSTRUMENT

Legend: Strongly Agree SA, Agree A, Undecided U, Disagree D, Strongly Disagree SD

The following questions deal with your perception of the undergraduate Computer Information Systems program at Bentley College. Please indicate the strength of your agreement/disagreement with each statement by circling the letter(s) that best describe your feeling about or reaction to each statement. The typical graduate of this program:

1. Is a whiz at mathematics. SA A U D SD
2. Entered Bentley College with a strong prior background in computers. SA A U D SD

During his/her first job after graduation, the typical graduate of this program:

3. Spends most of his/her working time writing computer programs. SA A U D SD
4. Spends most of his/her working time interacting with other persons. SA A U D SD
5. Spends most of his/her time working alone. SA A U D SD
6. Designs new computer hardware. SA A U D SD
7. Interacts mostly with other computer people. SA A U D SD
8. Has a starting salary above the average Bentley College graduate. SA A U D SD
9. Helps managers select new computer systems. SA A U D SD

The students who provided names and addresses in the first survey were used in the second survey eleven months later. The questionnaire was mailed directly to the students at their homes near the end of the summer break in August. This time the nine-item instrument was also contained within a larger questionnaire that additionally assessed computer anxiety. A $100 lottery was offered as an enticement to encourage responses.

RESULTS

Scale Properties

Dimensionality analysis was conducted on the nine-item instrument using the data from the Fall 1987 responses. The factor analysis and scale reliability analysis failed to yield sufficiently interpretable and reliable subscales. Consequently, as a result of these analyses, the nine items used to measure perceptions of workstyle were treated as separate items. [See (9) for a more detailed description of the subscale analysis performed.]

Stability of Student Perceptions

In order to determine whether the CIS workstyle perceptions of incoming freshmen were the same from one entering class to the next, the responses from the Fall 1987 survey were compared to the responses from the Fall 1988 survey. The standard deviations for the corresponding items were compared using F-tests and showed no significant differences. The averages for the corresponding items were compared using t-tests and also showed no significant differences. This would suggest the perceptions of incoming freshmen are reasonably stable from year to year.

Testing the Hypothesis

The means and standard deviations for the pre and post treatment responses to the nine items are listed in Table 2. Four items (#1, #5, #8, and #9) exhibited significant changes in their means with corresponding p-values of 0.01 or better. The score for Item #1 decreased which indicates stronger disagreement with the perception that the typical CIS graduate is a whiz at mathematics. The score for Item #5 decreased which indicates stronger disagreement with the perception that the typical CIS graduate spends most of his/her time working alone. The score for Item #8 increased which indicates stronger agreement with the perception that the typical CIS graduate has a starting salary above the average Bentley College graduate. The score for Item #9 increased which indicates stronger agreement with the perception that the typical CIS graduate helps managers select new computer systems.

TABLE 2. PERCEPTIONS OF CIS WORKSTYLES

	PRE		POST			
Item	Mean	S.D.	Mean	S.D.	t-val	Sig
1	1.70	0.98	1.47	0.89	3.20	**
2	1.67	0.99	1.55	0.97	1.56	
3	1.78	0.89	1.84	0.98	-0.69	
4	2.37	0.82	2.43	0.91	-0.70	
5	1.73	0.84	1.54	0.86	2.57	**
6	1.75	0.89	1.74	0.97	0.06	
7	2.19	0.95	2.17	1.03	0.21	
8	2.25	0.84	2.86	0.83	-8.68	***
9	2.38	0.82	2.63	0.76	-3.40	***

2-tailed:
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

DISCUSSION AND CONCLUSION

We think these results indicate reasonable success in correcting student misconceptions of the CIS profession. Specifically, we think much was accomplished to alter the "programmer image" evidenced by their responses to the first survey. The four items which showed a significant shift all changed in a direction that we think is favorable with respect to encouraging them to consider pursuing CIS careers.

As educators, we would like to think we have demonstrated that lasting changes can be made in students' perceptions with an exposure of only one class meeting period. However, the CIS careers session was not the only variable that might have influenced the observed result. There was another factor which we believe may have been at least as important. Prior to the beginning of the course it was established on a department-wide basis that a primary (although unwritten) objective of the course was to recruit students into the "gateway course" for the CIS major. This "gateway course" is a Pascal programming course which is normally taken in the second semester of the freshman year. In the previous year (1987-88) there were 21 freshmen who entered Bentley as declared CIS majors. In the subsequent spring semester the total enrollment in the "gateway course" was 31--a gain we think would have occurred regardless of the CIS careers session the previous semester. In the year of the current study (1988-89) there were 24 freshmen who entered Bentley as declared CIS majors. In the subsequent spring semester the total enrollment in the "gateway" course was 96--a very substantial increase which verified the success of the recruitment effort.

It is, of course, impossible to isolate the various events and relationships that occurred in the period spanned by the two surveys of the freshmen class of 1988-89. We know what the intended content of the CIS careers session was supposed to be, and we are also aware of most of the other activities in which

instructors engaged in order to recruit students. It is quite clear that the combined effect of these activities did result in changed perceptions of the CIS graduate's workstyle and also in a substantial increase in the number of students enrolled in the "gateway" course, but we cannot be at all certain which activity caused what result.

This study raises many issues--issues that we think need to be discussed in public forums. Given the continued decline in freshman interest manifested in the most recent UCLA report (5), it is imperative that action be taken to reverse that trend. Converting students from other majors is one such method that very obviously can be made to work. We need to go further to identify what specific actions and conditions are most supportive of this quest. A related issue that is at least equally important is the ethical question of using the classroom to try to "sell" a major and possibly abuse what we feel is a very special relationship between student and mentor. The implications for academic honesty and evaluation certainly need to be addressed. As successful as such a program might become within a given institution, we feel that the best place to focus action is at the high school level. Such actions need to effectively reach the four major constituencies involved in making college application decisions: students, parents, teachers, and guidance counselors. The problem is one of national importance and the scale of effort must be commensurate--involving not just the affected faculty, but also involving concerned business people and professional societies.

REFERENCES

1 Astin, A. W., Green, K. C. & Korn, W. S. (1985). The American Freshman: 20 Year Trends, American Council on Education and the University of California at Los Angeles.

2 Astin, et. al. (1986). The American Freshman: Norms for Fall 1986, American Council on Education and the University of California at Los Angeles.

3 Astin, et. al. (1987). The American Freshman: Norms for Fall 1987, American Council on Education and the &University of California at Los Angeles.

4 Astin, et. al. (1988). The American Freshman: Norms for Fall 1988, American Council on Education and the University of California at Los Angeles.

5 Astin, et. al. (1989). The American Freshman: Norms for Fall 1989, American Council on Education and the University of California at Los Angeles.

6 Cale, E. G., Mawhinney, C. H. & Callaghan, D. R. (199x). Student perceptions of information systems careers: misconceptions and declining enrollments. To appear in the Journal of Research on Computing in Education.

7 Couger, J. D. & Zawacki, R. A. (1980). Motivating and Managing Computer Personnel, John Wiley & Sons, New York.

8 Duke, P. (1987). Jobs go unfilled as fewer students show interest in computer science. Wall St. Journal, November 27, p. 13.

9 Mawhinney, C. H., Callaghan, D. R. & Cale, E. G. (1989). Modifying perceptions of the CIS Graduate's Workstyle. ACM-SIGCSE Bulletin, 21(1), February, pp. 78-82.

10 Mawhinney, C. H., Cale, E. G. & Callaghan, D. R. (199x). Freshman expectations of information systems careers versus their own careers. To appear in CIS Educator Forum.

11 Rochester, J. B. (1988). The crisis in computer education. Computer Update, January/February.

12 Whitten, J. L., Bentley, L. D., & Ho, T. I. M. (1986). Systems Analysis & Design Methods. Times Mirror/Mosby, St. Louis.

13 Withington, F. (1988, January 11) Only you can prevent an MIS shortage. Computerworld, January 11, p. 17.

14 Wood, M. (1979). Systems analyst title most abused in industry: redefinition imperative. Computerworld, April 30, pp. 24, 26.

IS STRATEGIC PLANNING FOR BUSINESS EDUCATION PROGRAMS

Harry C. Benham
College of Business
Montana State University

Abstract

The paper uses a case study of Information Systems Strategic Planning in a Business College environment to argue that substantial benefits can be obtained from such planning. Techniques for aligning IS strategy with the College's strategy and for obtaining top management's commitment are covered. Details of assessing the current environment, identification of fundamental problems, recommendations, and the implementation plan are given. Finally, the expected and unexpected benefits of the plan are enumerated.

In a business setting, the benefits of Information Systems Strategic Planning (ISSP) have been widely identified (King, 1978). Indeed, Brancheau and Wetherbe (1987) found strategic planning to be the most critical issue facing IS executives. By relating one business college's experience with ISSP, this paper argues that business colleges can experience substantial benefits from ISSP.

Traditionally, ISSP benefits derive from identification of strategic opportunities, alignment of IS with the organizations goals, and the systematic, rational allocation of resources.[1] Strategic opportunities exist and can be identified within a business college ranging from the positioning of a specific degree program to the utilization of IS to enhance students' communication skills. An information system designed to support a school's research and teaching objectives will facilitate attainment of those goals. Finally, capital funds are notoriously short within academic institutions. An ISSP provides college administrators with a reasonable view of IS funding requirements into the future.

In addition to traditional ISSP benefits, an ISSP provides additional potential benefits within a business college. First, the appropriate academic "home" for IS continues to be an issue within business colleges. An ISSP provides a framework within which this issue can be addressed. Second by demonstrating that the IS domain extends beyond COBOL programming, an ISSP provides an opportunity to enhance an IS program's image as an academic discipline.

The ISSP began with selection of a methodology from the wide array of IS planning methodologies currently available[2]. A variant of IBM's Business Systems Planning methodology was followed. This paper's organization reflects the major phases in the methodology applied: Setting the ISSP Goal and Objectives, Assessing the Current Environment, Problem Identification, Recommendations, and the Implementation Plan. The final section of the paper reviews the benefits derived from this ISSP exercise.

SETTING THE ISSP GOAL AND OBJECTIVES

Setting of the ISSP goal and objectives consisted of two parts. The first part was assessment of the overall university, college, and campus computing agendas. At the business

college level, there were three "organizations" to consider, the university, the business college, and the university's central computing. To some extent, existing documents provided information on the planned strategies of these organizations. Interviews with top university officials, the college dean, and the director of central computing completed the picture.

The second part consisted of drafting a goal and enumerating specific objectives. The ISSP goal focus directly on support of the business college's agenda consistent with university and central computing plans. Objectives were specific focal areas which contribute to attainment of the overall ISSP goal. The draft goal and objectives where presented to the college dean and the director of central computing for comment. The finalized ISSP goal and objectives included:

Goal:

To develop an integrated computer information processing and management environment that synergistically supports the College of Business mission and priorities.

Objectives:

(1) Provide the capability for high use of computer aided delivery of instruction.

(2) Provide students with convenient access to a computing environment which fosters development and use of analytical and communication skills.

(3) Provide a computing environment that is representative of the one students will use in a modern business world.

(4) Provide computing facilities, services, and support for faculty, staff, and student research and service in all areas of business.

(5) Provide infrastructure to ensure adequate staffing, operating, training, and maintenance.

Those participating in this ISSP were well aware that alignment of the IS strategy with the organizations strategy and top management commitment have been identified as ISSP critical success factors.[3] By direct design, the ISSP goal and objectives were to align IS capabilities with the organizations plans. Direct involvement of "top management" in eliciting the organizations strategy and their review of the ISSP goal and objectives as they were being formulated provided the foundation for their ultimate support and commitment to the ISSP.

ASSESSING THE CURRENT ENVIRONMENT

In assessing the current environment, information was gathered from four groups: faculty, students, staff, and employers. Faculty, students, and staff were asked about their current computer uses, problems they encountered, and potential uses. With clearly identified ISSP objectives, it was possible to structure questions to elicit whether or not each objective was being met. In those cases where the objective was not being met, the respondent often provided insight into why it was not met.

In gathering this information, wide coverage and detailed specifics were desired. To obtain wide coverage, all faculty and staff were encouraged to complete questionnaires and a representative sample of students were asked to complete questionnaires. To obtain detailed specifics, interviews were scheduled with selected faculty, students, and staff. The selection process insured that some representative for every identifiable group was interviewed. In addition, any individual who displayed a desire to provide input beyond their questionnaire responses was interviewed.

Employers of graduates from each business college degree granting program were contacted. These individuals were asked to provided their assessment of the adequacy of computer and other skills exhibited by graduates. Although most employers expressed satisfaction with the graduates' skills, there were exceptions. Notably, those critical of the IS knowledge possessed by our graduates had hired in non-IS disciplines.

IDENTIFICATION OF PROBLEMS

The information gathered from faculty, students, staff, and employers was reviewed and synthesized into a relatively short list of fundamental problems. The following anticipated fundamental problems were identified:

(1) Hardware: An acute deficiency of computers and computer related equipment was found to limit computer aided instruction, MIS instruction, student computer use, and research.

(2) Facilities: Laboratory space constraints, classroom equipment, and electrical wiring were found to limit computer usage.

(3) Communications: Inadequate communications among computers and computer users prevented effective sharing of equipment and information.

The following unanticipated fundamental problems were identified:

(1) Support: Inadequate consulting, training, and maintenance services prevented effective utilization of existing computing resources.

(2) Literacy: A lack of computer skills, particularly micro-computer skills severely limited computer usage at levels.

(3) Data Access: Access to business related data was inadequate and inconvenient prohibiting its instructional use and hindering its research use.

Whether the problem was anticipated prior to the ISSP or identified during the ISSP, the ability to clearly document each problem and its manifestations was valuable. Consider for example the lack of computer literacy problem. This problem appears to be a major inhibitor of integrating computer applications into the curriculum. From the faculty questionnaires and interviews, it was discovered that many faculty were reluctant to substitute course content for, in their view, remedial computer instruction. Further from the faculty and student responses, it was clear that computer instruction was required in order to effectively integrate a computer application into a course's content. Yet when the necessary computer instruction was provided, those few computer literate students were appalled by class time being devoted to elementary skills training.

RECOMMENDATIONS

The recommendations phase of the ISSP consisted of finding a set of recommendations which would simultaneously solve all identified fundamental problems. The recommendations included the following:

(1) Information Center: The concept of an information center employed here is as single source provider of software, hardware, data access, and communications "support" where support includes consulting, training, technical trouble shooting, and maintenance.

(2) Curricular Changes: Topics in computer literacy, with an emphasis on micro computer literacy, should be introduced as early as possible in the curriculum.

(3) Additional Administrative Policies: Administrative policies focused on increased standardization and training, maintenance and replacement, and on-going IS planning were needed to insure that the college's IS resources continue to support the college's priorities.

(4) Expanded Computer Labs: Increase the number of computer Labs to support basic skills training, IBM microcomputer applications, Macintosh microcomputer applications, mainframe access, and general purpose student access.

(5) Classroom Upgrades: Equip classroom with multi-media projection capabilities and connection to computer network.

(6) Install a Mini-Computer: Functions of a mini-computer would include provision of a "mainframe environment" for IS degree program instruction, flexible communications control, business data access, and research computing capacity.

The first three recommendations were targeted at the literacy and support problems. By placing computer literacy instruction as early as possible in the curriculum and establishing an organization to provide training and software consulting services students would be able to acquire and enhance the computer skills they would need. In addition to faculty, student, and staff support, the information center would be responsible for maintaining a business database. The additional policy recommendation is an acknowledgement that IS planning should be a continuous process including hardware replacement[4]. The remaining three "equipment" recommendations respond to the anticipated problems of hardware, facilities, and communications. The only novel aspect of these recommendations is the choice of a mini-computer based network over a local area network. One factor in this choice was the view that a mini-computer was required to provide a "mainframe" environment for IS instruction. Another factor was the access to business and financial data requirement. The volume of this data and its anticipated usage, would tax the capabilities of a local area network.

IMPLEMENTATION PLAN

Recent research suggests that the implementation plan phase of an ISSP is critical to the plans success[5]. The five phase implementation plan described here reflects two basic principles: the organizational support function should be in place before major hardware changes and those recommendations yielding the most benefit for resources expended should be implemented first. The first phase, lasting only three months, called for instituting the required curricular changes and revising the college's organization to accommodate an information center. The second phase, lasting nine months, called for the founding of the information center and upgrading and/or replacing current PC lab equipment.

The third phase would see the major ISSP recommendations implemented. By the end of the eighteen month phase, the information center would be fully functional, the mini-computer would be installed and networked to all college microcomputers.

The fourth and fifth phases, at twelve and eighteen months respectively, would see the addition of "high-tech" classrooms and laboratory facilities.

BENEFITS

A number of benefits of implementing the ISSP recommendations were identified in the ISSP. The most important of those benefits were:

(1) Excellence in teaching through

increased professionalism and innovation,

(2) Maintenance of industry expectations concerning the computer skills of the business college's graduates, and

(3) Assisting in the attraction and retention of top quality faculty through the provision of computing resources which facilitate their research activity.

Additional ISSP benefits have accrued since the plans adoption. Simply having a "plan" has been instrumental in dealing with central computing and the university's central administration. The "plan" has also attracted vendor and donor interest. Vendors, obviously interested in having their equipment utilized, have donated demonstration equipment and assisted with locating funding sources. With an established plan, donors seem encouraged by the knowledge of how their contributions will fit and further the college's overall plan.

Finally, the ISSP seems to have enhanced the academic standing of the IS faculty. By putting their classroom theories into practice to provide benefits for the entire college, the IS faculty have demonstrated that their discipline's domain extends well beyond the training of programmers.

REFERENCES

(1) Brancheau, J.C. and J.C. Wetherbe, "Key Issues in Information System Management," MIS Quarterly, March, 1987, 11(1), 23-42.

(2) Davis, G.B. and M.H. Olson, Management Information Systems, 2nd Ed., McGraw Hill, 1985

(3) Dickson, G.W. and J.C. Wetherbe, The Management of Information Systems, McGraw-Hill, 1985.

(4) King, W.R., "Strategic Planning for Management Information Systems,MIS Quarterly, March 1978, 2(1), 27-37.

(5) King, W.R., "Strategic Planning for Management Information Resources: The Evolution of Concepts and Practices", Information Resource Management Journal, Fall 1988, 1(1), 1-10.

(6) Lederer, A. and V. Sethi, "Implementation of Strategic Information Systems Planning Methodologies," MIS Quarterly, Sept. 1988 12(3), pp. 445-461.

(7) _____, "Critical Dimensions of Strategic Information Systems Planning," Decision Sciences, (forthcoming).

NOTES

1. Davis and Olson, (1985) pg. 446.
2. Dickson and Wetherbe, (1985).
3. Lederer and Sethi, (1988).
4. King, (1988).
5. Lederer and Sethi, (forthcoming).

The MIS Curriculum: Which Competencies Are Really Important to Business Professionals?

Thomas A. Pollack
Duquesne University

ABSTRACT

Many technical competencies are embedded in the courses of the model curricula for information systems advocated by the DPMA and ACM. The development of an arsenal of technical competencies is significant for a student preparing for a career in the information systems field. A survey of the members of the MIS Advisory Board of Duquesne University revealed that this select group of corporate officials is looking for more than technical ability in its recruits. The ability to communicate and a solid commitment to ethics and values are deemed to be as equally desirable as technical competence in the opinion of our MIS Advisory Board. The task which we, as educators, face is to devise a way to effectively factor all of these components into our MIS curriculum.

INTRODUCTION

A great deal has been written about Management Information Systems (MIS) curriculum offerings in colleges and universities across the country. Invariably, discussion on the topic will focus, in some way, on the Data Processing Management Association (DPMA) (3) and Association for Computing-Machinery (ACM) (1) curriculum guidelines for information systems education. A common question is "How closely does your program follow the model curriculum?" Certainly, the guidelines provided in the "model curricula" have proven to be valuable to Information Systems educators. If nothing else, model curricula initiate healthy discussions concerning what should not be present in a program of studies. Many philosophical points of view and differences arise as a result of this discussion. All of this is important to an evolving discipline such as MIS. The truth is that most faculties obtain ideas from model curricula, research efforts, conferences, and peer discussions. These ideas are then synthesized into a cohesive set of courses to comprise a curriculum.

It is obvious to even the most casual observer that MIS has not standardized its offerings as a discipline as is the case with a discipline such as accounting. When someone says "Cost Accounting," all who are familiar with the accounting discipline can relate rather readily to course content. On the other hand, the same cannot be said of "Computer Hardware/ Software Systems." The commonality of course content and title which exists in accounting does not carry over to MIS. Actually, there are many who refuse to recognize MIS as a discipline. So it is, that MIS, although relatively popular in schools of business, is a discipline that continues to struggle to establish a true identity. The dynamic nature of the fields of computing and systems has not helped a great deal in stabilizing the discipline.

Because of all the controversy which exists concerning MIS curricula, from time to time, I have attempted to identify important competencies and validate the overall significance of our curriculum offerings with the regional

business community. This paper describes one such effort which was carried out in conjunction with the members of the Duquesne University MIS Advisory Board.

MIS CURRICULUM

To say that there is a general lack of consensus as to what should constitute the total curriculum which prepares future MIS professionals is an understatement. In recent years, we in higher education have shown improvement in that we have reached relative consensus on a "common body of knowledge" for MIS.

The need for adequately trained MIS professionals in the workplace is increasing as organizations become more dependent upon information systems. (12) Many practitioners and educators feel this creates a problem because of the general inadequacy of MIS curricula to properly prepare future professionals. To compound the problem, a recent article in the New York Times confirmed what many of us already know: although there is a growing demand for entry level and advanced computer professionals, the number of students majoring in computer-related fields has declined. (5) What all of this means is that, we as IS educators, must constantly assess our programs and seek reactions to the preparedness of our product, namely the students who graduate and are employed in our local business communities. We must strive to be better.

The literature is full of suggestions about what should happen with MIS. These suggestions range from total integration within the business disciplines as opposed to specialized programs in MIS (8) to a concentrated emphasis on end-user computing within the MIS curriculum. (4) Some IS researchers have used the model curricula published by DPMA and ACM as "springboards" for new ideas and curriculum innovations. (10) Everyone has ideas about what to do with MIS. Everyone has ideas about what the curriculum should contain, and everyone has ideas about what the curriculum should accomplish. One thing is certain: in MIS, we are dealing with a very dynamic field. An institution cannot become complacent in revision efforts dealing with the MIS program of studies. Not only must the MIS curriculum reflect what is currently state-of-the-art, it must project what will be state-of-the-art five years hence.

At Duquesne University we have made a concerted effort to maintain a collaborative relationship with the Pittsburgh business community and the MIS professionals in particular. We have a very active MIS Advisory Board, and we have come to rely on our advisory board to provide feedback on a number of things that we do. Curriculum feedback is one of those areas which we deem to be very important. We want the advisory board to validate what we attempt to accomplish or to refute some of what we do as being unimportant. That is precisely why we approached our captive audience advisory board and sought their assessment of importance for a number of competencies which we are presenting or can present in our curriculum. Hopefully, by sharing these opinions, some of your thinking will be validated or refuted and perhaps you will find that some of these opinions are simply thought-provoking.

SURVEY

The survey instrument used for this paper has been updated and revised over a period of years to reflect current commonly used hardware and software. (9) Most of the competencies listed in the survey are covered in our MIS program of studies. The complete questionnaire is included in APPENDIX A.

Although the questionnaire will be widely circulated among both MIS educators and the business community, an

initial solicitation of responses was sought from the members of the MIS Advisory Board of Duquesne University. The Advisory Board is comprised of sixteen very high ranking information officers from fifteen major corporations. For this reason, the opinion of the members of the advisory board is both meaningful and significant to us. It will be interesting to see if the responses of this "captive audience" correlates with other members of the general corporate community as distribution of the questionnaire is expanded.

The results which are presented in this paper are the opinions of fourteen high ranking information officers from major Pittsburgh-based corporations. A total of sixteen questionnaires were distributed, and thirteen were completed. Although this represents a small sample, it represents a very creditable set of responses. It represents the opinions of high-ranking officials who are very much in tune with today's information systems needs and who are very familiar with MIS in higher education. The respondents occupy responsible positions which permit them to dictate the type of individuals who will be hired by their organization.

The major components of the survey and the average responses from our selective survey group are presented for your examination. A rating of 1 is Least Important and 5 is Most Important. The mean and standard deviation for each major response category is presented.

SURVEY RESULTS

MIS Competencies

	Mean	Std Dev
1. Structured Anal and Design	3.29	0.82
2. Structured Program Techniques	4.00	0.55
3. Project Mgmt	3.07	0.61
4. File Organization and Design	3.84	0.76
5. Documentation and Maintenance	3.30	0.99
6. Mainframe Operating Sys	2.53	1.00
7. IBM Job Control Language	3.54	1.08
8. Other JCL	1.92	0.91
9. Microcomputer Operating Sys	2.76	1.18
10. Database Management Systems	3.92	0.82
11. Data Modeling	3.07	0.72
12. Computer Simulation Models	2.23	0.79
13. Artificial Intell/ Expert Systems	2.30	1.06
14. End-User 4GL (FOCUS, etc.)	3.30	0.91
15. Appl Generator 4GL (Oracle, etc.)	3.23	0.69
16. Decision Support Systems	2.61	0.83
17. Computer Graphics	2.46	0.84
18. Data Communications	3.23	1.18
19. EDP Auditing	1.84	0.76

Software

20. MVS-TSO	3.76	1.12
21. CICS	3.07	1.14
22. UNIX	2.38	1.07
23. VM-CMS	2.07	0.99
24. VMS	2.00	0.96
25. LAN System Software	3.00	1.24
26. IMS (Hierarchial)	3.15	1.02
27. SQL (Relational)	3.69	0.91
28. IDMS (Network)	2.15	1.29
29. COBOL	4.69	0.60
30. FORTRAN	2.00	1.24
31. BASIC	2.38	1.07
32. Mainframe Assembler	2.15	1.34
33. C	2.83	1.40
34. FOCUS	2.69	0.82
35. ORACLE	2.23	1.24
36. NOMAD 2	1.38	0.48

Artificial Intelligence Languages

37. LISP	2.00	0.87
38. PROLOG	1.76	0.57

Expert System Languages

39. EXSYS	1.54	0.65

40. 1st Class	1.81	0.93
41. Level V	1.81	0.93
42. TI Personal Consultant Easy	1.54	0.65
43. Neural Networks	1.45	0.65
44. VP Expert	1.54	0.65
Statistical Packages		
45. SAS	3.46	0.84
46. SPSS	2.53	1.00
47. IFPS	2.55	0.95
48. PSG	1.75	0.82
Financial Modeling		
49. Spreadsheets	3.30	1.20
50. Micro Database	3.00	1.17
51. Word Processing	3.07	0.99
Other Competencies		
52. Written Communication	4.46	0.63
53. Oral Communication	4.30	0.46
54. General Business Theory	3.53	0.74
55. Management Principles	3.15	0.66
56. Accounting Principles	3.07	0.72
57. Finance Principles	2.69	0.60
58. Marketing Principles	2.30	0.91
59. Production Mgmt	2.38	1.14
60. Group Dynamics	3.61	0.73
61. Organizational Behavior	3.15	0.53
62. Interpersonal Skills	4.23	0.57
63. Ethics/Values	4.53	0.63
64. Liberal Arts Core	3.00	1.17

SURVEY ANALYSIS

To the chagrin and dismay of many, our respondents ranked COBOL knowledge as the primary technical competency for an MIS graduate to possess. In order of importance, they ranked Ethics and Values next. This is not surprising in that business ethics is certainly getting a great deal of attention from the business community at this time.

Our School of Business and Administration is currently undertaking a project to integrate units on ethics across all business disciplines. In a recent article in the Information Executive, Bruce Spiro states:

> "IS professionals may not have all the answers, but certainly they have more experience with ethical issues than others. ... If the clergy and parents aren't equipped to teach ethics, I suggest that our schools, with assistance from information executives, can. The concepts of information value, intellectual property and the rights of privacy must be taught and continually reinforced. (11)

Spiro goes on to emphasize the importance of teaching ethics as a standard in all of our nation's schools. Most importantly, he indicates, is that professionals from other disciplines must also be part of the emphasis on ethics. (11)

At or near the top of the list, each time I have administered this survey, are written and oral communications and interpersonal skills. This whole area speaks for itself, as over the years, numerous criticisms have been leveled against higher education for not developing better communication skills. Based on survey results, it is obvious that the ability to communicate is deemed to be one of the vital keys to success in the business environment.

In that this group of survey respondents identified COBOL as a very desirable competency, it is not surprising that they also ranked knowledge in the area of structured programming techniques and database management systems rather highly. Most entry level positions in the large corporations involve some form of programming. Naturally, the more knowledge-able an individual is in this

area, the more productive that individual can be as an entry level employee. The list of competencies which are deemed most desirable also contains knowledge of operating systems, both mainframe and microcomputer, and structured analysis and design techniques. The majority of respondents are from IBM environments.

The most highly rated competencies represent a blend of technical, communications and value-oriented qualities. From this, it is safe to conclude that MIS, like other disciplines, is seeking individuals with well-rounded educational backgrounds for entry level positions. This provides further evidence that the day of the pure "techie" has long ago passed us by.

Ranking low in importance on the list of desired competencies is the 4GL, NOMAD 2, and the entire group of expert system languages and expert system development tools. This notion is contrary to the current impetus in business to develop expert systems. Most of this developmental work is apparently taking place within small sub-groups of the organization with a handful of people currently involved. Rounding out the list of least important competencies are EDP Auditing and Computer Simulation Models. Like model curricula, compiled survey results such as this initiate excellent philosophical discussions. There is no single "right way" to proceed in MIS education. Some of the most healthy curriculum ideas emanate from discussions among professional scholars with a common interest. Survey results always represent the opinion of the sample used. In this case, the sample is a small, but selective group of Information System professionals. The author places a high value on the opinions expressed by this group, but there is good reason to seek input from a larger sample.

CONCLUSION

Based on the outcome of the MIS Advisory Board survey respondents, several conclusions can be drawn. First, academic institutions must emphasize the overall development of communication skills. Students must be critiqued on writing, speaking, and interpersonal skills as well as group dynamics. Secondly, business ethics and values should become an integral part of the curriculum. Many in the business community feel that additional emphasis should be given to this area. Thirdly, academic institutions must continue to develop technical competency and problem-solving skills. In most creditable institutions, the proper blend of technical competence will fall into place, perhaps more naturally than the previously-mentioned competencies.

Academia also must assume a forward-looking role. A closer examination of the survey responses lead me to the conclusion that the Advisory Board respondents focused more on what technical competencies are important to be productive immediately. They did not attempt to look five years hence and predict what competencies might help to create a strategic, competitive edge. To substantiate this statement, allow me to direct your attention to the survey responses in the entire artificial intelligence/expert systems section. These competencies are not deemed important, in a realistic sense, by the respondents. However, the literature, in general, indicates that this is a very important area for the future.

MIS cannot be all things to all people. It will continue to be a controversial discipline, unstructured in comparison to a discipline such as accounting. Questions remain unanswered and problems remain unsolved. As MIS educators, we must continue to progress and move forward if we expect our business schools to properly train the managers who will lead the nation's business into the next century.

REFERENCES

1. ACM Curriculum Committee on Information Systems Curriculum Recommendations for the 80's "Undergraduate and Graduate Program," New York, Association for Computing Machinery, 1983.

2. Chen, Jim and John A. Willhardt. "Computer Curricula in AACSB Accredited Business Schools (1987)," Interface, Spring 1988, pp. 28-31.

3. Data Processing Management Association, "CIS '86 -- the DPMA Model Curriculum for Undergraduate Computer Information Systems," Second Edition.

4. Day, John C., Anne H. McClanahan, and James L. Perotti. "Incorporating End-User Computing Into an MIS Curriculum," Interface, Spring 1989, pp. 50-52.

5. Fowler, Elizabeth M. "Computer Job Demand Up, Interest Off." The New York Times. February 13, 1990, p. C17.

6. Hartog, Curt. "Of Commerce and Academe," Datamation, September 1, 1985, pp. 68-78.

7. Jenkins, George H. "Education Requirements for the Entry Level Business Systems Analyst," Journal of Systems Management, August 1986, pp. 30-33.

8. Nazem, Sufi M. "Business Schools and MIS Curricula: The Need for Integration, Interface, Fall 1987, pp. 66-69.

9. Pollack, Thomas A. "The Rationale, Design, and Assessment of a Business Information Systems Curriculum to Fulfill the Needs of Large Computer Users," Dissertation presented to the University of Pittsburgh, 1981.

10. Redmond, Richard T. and A. James Wynne. "The Changing Requirements of Information Systems Education and Its Impact on Undergraduate Curriculum," Interface, Fall 1988, pp. 38-45.

11. Spiro, Bruce E. "Ethics in the Information Age," Information Executive, Fall 1989. pp, 38-41.

12. Yaffe, Jerry. "MIS Education: A 20th Century Disaster," Journal of Systems Management, April 1989, pp. 10-13.

INTEGRATED SYSTEMS SPECIALIST: A COMPREHENSIVE CURRICULUM APPROACH

Jean Buddington Martin
Douglas Kerley
L. Gail Lefevre

Florida Community College at Jacksonville
Jacksonville, Florida

ABSTRACT

The Integrated Systems Specialist associate degree program described in this paper was developed in response to the tremendous expansion of microcomputers and the resultant need for trained professionals with skills in using more sophisticated software packages in a networked environment. We looked at the proliferation of microcomputers within our college and realized that we are a microcosm of the business community throughout our city and the entire nation. A review of the literature disclosed a growing emphasis on end-user computing. Training is an essential ingredient for success in this environment, but who trains the trainers? Computer hardware and software are becoming too complex for anyone to become a self-trained expert. The Integrated Systems Specialist degree program teaches a student the set of skills needed to support and train users of microcomputer systems. An Integrated Technology Center has been developed to support the program with a place for students to practice newly acquired skills. The concept of the ITC is a dynamic "living laboratory" which will serve as a demonstration center by integrating the latest technology in product design and manufacturing with marketing, accounting, information systems and office technology. We see the Integrated Systems Specialist degree supported by the Integrated Technology Center as a curriculum for the 1990's.

CONCEPT DEVELOPMENT

One of the major attributes of community colleges has been their ability to respond to the varied needs of their constituencies in a timely manner. "Community colleges must be 'adaptive, and creative'. They must maintain their ability to identify and to move quickly to address the changing needs of society as a whole as well as the needs of their students" (Davenport, 1989 p. 26). Nowhere is the ability to be adaptive and creative more necessary than in the field of computing. Computer information systems faculty at many community colleges nationwide are endeavoring to develop new programs or modify old ones in order to more closely mirror the needs of their constituents. Revisions to the traditional programming and analysis programs certainly provide a clear challenge, but that is not the only challenge. The importance of the need for skills in microcomputer technology (software and hardware), networking, communications, and human relations are equally apparent. The Associate Degree described in this paper is related to this set of skills and is an addition to our traditional computer and information system degrees.

As a result of a presentation of an internal self assessment, which is conducted on a 5 year schedule at the Florida Community College at Jacksonville, our community college President mandated that the CIS discipline determine the direction to be followed in the foreseeable future. Fortunately there were a number of sources to consult at each developmental stage. A recent advisory committee CIS survey report provided valuable initial input. Additional insight was gained through observing one type of computer skills that were presently in great demand at our own institution. We were hiring people to support the many computer labs that had been installed. We also needed support for the tremendous proliferation of computers that were being ordered and used by faculty, administrators, professionals, and staffers throughout the college. Along with this tremendous expansion of microcomputers was the need for greater skills in using more sophisticated software packages. Networking of microcomputers within the various labs and offices throughout the college were already underway, and plans are being made to extend the communication environment so that eventually everyone at the college, on every campus, can be connected to all computing resources. The evolution at our college is right on target with the trends reported by Peter Van Cuylenburg, vice president of the Data Systems Group for Texas Instruments (1989). Many more advanced uses of the technology can be expected as we proceed into the decade of the '90s. When we looked outside our institution we realized that what was happening at our college was also happening in businesses throughout the community that we serve. As might be expected, their computing needs were not unlike ours.

A review of the literature disclosed a growing emphasis on end-user computing (Beccue & Chrisman, 1987; PC World, 1989; Kerr, 1989; and Van Cuylenburg, 1989;) Responses from a Datamation survey of 4,700 users revealed "... a number of far-reaching trends likely to shape the course of the U.S. computer industry-and perhaps the word stage-in 1990 and beyond" (Francis, 1989 p. 29). A sampling of the trends include spectacular growth in the U.S. PC market; continued growth in networks and associated applications; expanded use of multiple PC suppliers for single user organizations; and, although MS/DOS will continue to hold its ground against OS/2, big gains are forecasted for UNIX and RISC architecture.

Training is an essential and inherent ingredient for success in the forecasted environment. Where do the trainers come from? Who will design, purchase, and maintain these systems? How is an end-user support person trained? In the past much of the training was done informally by the in-house, often self-trained guru. This amateur has become the office expert! Unfortunately, the results were sometimes far from ideal. Espy and Howe ask, "Must well-meaning professionals suffer the confusion and intimidation of so-called experts without a whimper" (1989, p.103). If the support person is properly trained this type problem can be minimized. Also, equipment and software are becoming too complex for someone to become a self-trained expert.

SKILL CATEGORIES

After reviewing the literature and completing our preliminary survey of community needs, a rough outline was developed which gave an overview of the type of skills that we had identified as being critical. A more in-depth analysis was then conducted to refine and expand on the skill areas that had been identified. This was carried out through interviews with representatives from local businesses as well as key in-house IS personnel. Having an extensive understanding of the skills and skill levels necessary was considered crucial to the program development process.

Curriculum development was not undertaken until this step was satisfactorily completed.

The five identified skill categories included:

I. Written Communication
II. Oral Communication
III. Interpersonal Skills
IV. Analytical Skills
V. Technical Skills

These categories were further divided as follows:

I. Competency in written communications, such as
   A. Developing specifications for the purchase of hardware and software.
   B. Putting together "Requests For Proposals"
   C. Preparing system documentation.

II. Effective oral communication with
   A. Company management
   B. Equipment repair personnel
   C. Peers within the profession
   D. All company personnel who use computers

III. Proficient interpersonal communication affording
   A. Productive computer training for company personnel
   B. Effective work with computer users in a problem solving mode
   C. Enhanced productivity as part of a team

IV. Well developed analytical skills leading to
   A. Troubleshooting hardware and software problems
   B. Increased problem solving ability
   C. Writing software in a fourth generation language

V. Competent technical knowledge allowing
   A. Evaluation of hardware and software for purchase
   B. Maintenance of a microcomputer network with a mainframe connection
   C. Currency with state-of-the-art technology.

INTEGRATED SYSTEM SPECIALIST CURRICULUM

Once the desired skill categories were fully understood, the curriculum was derived. Input from a variety of faculty was included in this stage of the development process. The final iteration of the curriculum is shown in Table 1. There are four new courses.

Introduction to Data Communications

The student will be introduced to the basic terminology of data communications. Topics will include fundamentals and concepts of transmission paths and data line controls/line protocols, major components in a data communication system and major components in a local area network.

Computer Networks

The student will study Local Area Networks (LANs), Wide Area Networks (WANs), PC to Mainframe connections, LANs gatewayed to mainframe hosts and other distributed systems. Hands on experience in generating and maintaining a LAN will be an integral part of the course. The skills taught in this course include those of a network administrator.

User Support and Software Evaluation

This course will present a host of skills critical to the success of an Integrated Systems Specialist. They will include: training end users to use PCs and PC software; installing software; dealing with printer problems; software and hardware evaluation; writing RFPs; hard disk administration; computer security; etc.

Internship
The student will work at a business in their area of interest.

Under "Professional Electives", two related courses will be chosen in an area of interest to the student. These courses, with the approval of the program advisor, allow the student to focus on a particular business or field of interest in which the student plans to work. For example, a student planning to work in a general business environment might choose to take Accounting I and II. Or, the student interested in programming could take two computer languages. Other possibilities include, but would not be limited to, introductory courses in Engineering, Law, Graphic Arts, Criminal Justice, Hospitality Management, Insurance, Radio/Television Broadcasting or the Health-related field.

CONCEPT OF A SUPPORT FACILITY

The curriculum development process highlighted the need for a special place or center to support the Integrated Systems Specialist program. Further investigation revealed that many of the college's technical/career and academic programs could benefit from such a facility. A center would provide an ideal setting for students in the newly designed program to practice their skills. Further, a center would provide the ability to demonstrate and practice the **integration** of various disciplines using the hardware and software of multiple vendors.

Again, extensive interviewing of key personnel was carried out in order to better understand the requirements of the various potential internal user groups. An evaluation of the types of organizations residing in our community was also undertaken. We wanted to be assured that the center we developed would also be able to serve local business. This exercise was combined with visits to existing technology centers at other institutions. Part of the task became recognizing what might work at our college as well as what would not be appropriate. Many articles were also read in order to broaden our comprehension of the directions that we might choose in the development process.

We were finally able to arrive at a definition for what has come to be known at the Integrated Technology Center (ITC):

> The concept of the center is a dynamic "living laboratory" which will integrate the latest technology in product design and manufacturing with marketing, accounting, information systems and office technology. It will serve as a demonstration center for its clients: 1) college students and faculty, 2) college staff, and 3) the local business community.

The mission of the center will involve the accomplishment of the following tasks:

1. The ITC will provide support to many technical/career and academic programs of the college.
2. The ITC will position the college for a flow of resources in grant money and equipment donations.
3. The ITC will establish business partnerships.
4. The ITC will serve as a test site for the high tech hardware and software of multiple vendors/partners.
5. The ITC will provide opportunities for its clients to use state-of-the art technology in an integrated setting.
6. The ITC will provide high profile and prestige for the college and its business partners.

SUMMARY

The development of the Integrated

Systems Specialist program has inadvertently allowed us the opportunity to take a broader more comprehensive look at the diversified computing field. We think that we are planning more wisely the use of our resources and looking more closely at how those expensive resources can serve multiple user groups. The development process has also allowed diverse faculty to better understand how their specialty can be integrated with other specialties so that everyone benefits, particularly the students.

References

Beccue, B., & Chrisman, C. (1987, October/November). Preparing students for careers in information centers. Proceedings of the sixth annual information systems education conference, San Francisco, CA. 55-58.

Davenport, L. F. (1989, February/March). The role of the community college in meeting America's future labor force needs. AACJC Journal, 23-27.

Espy, J., & Howe, J. (1989, December 4). Battling guru hoodoo. Computerworld, 23(49), 103-107.

Francis, B. (1989, November 1). The Desktop Dimension. Datamation, 35(21), 28-40.

Staff. (1989, Summer). Computing into the '90's: An interview with Peter Van Cuylenburg. Information Executive: The Journal of Information Systems Management, 2(3), 64-69.

Table 1. Integrated Systems Specialist Curriculum

General Education	Credit Hours
English Composition	3
Fundamentals of Speech Communication	3
Intermediate Algebra	3
Dynamics of Behavior	3
Humanities	3
	15

Professional Requirements	Credit Hours
Introductory Computer Concepts	3
Introduction to Programming and Algorithm Design	3
Spreadsheet Concepts and Practices	3
Database Management Concepts and Practices	3
Desktop Publishing	3
Microcomputer Operating Systems Concepts	3
Computer Systems Development W/High Level Tools	3
PC Troubleshooting	3
Word Processing I	3
Business Communications	3
*User Support and Software Evaluation	3
*Introduction to Data Communications	3
*Computer Networks	3
*Internship	3
	42

Professional Electives	Credit Hours
Area of Interest (Courses approved by the program advisor)	6
Total Semester Credit Hours:	63

* New Courses

AN ASSESSMENT OF THE ATTITUDES OF GRADUATES AND EMPLOYERS TOWARD COMPETENCIES NEEDED FOR ENTRY-LEVEL MIS POSITIONS

Mary Sumner
Robert Klepper
Robert Schultheis
Southern Illinois University at Edwardsville

ABSTRACT

The purpose of this study was to assess the attitudes of graduates and employers toward competencies needed for entry-level MIS positions. A total of 740 alumni and 170 employers were surveyed.

The findings showed that both graduates and employers believe that communications skills, systems design skills, and database design skills are important for successful job performance. The major difference between graduates' and employers' attitudes was in the area of programming skills. Employers' perceptions of the importance of programming skills were significantly higher than the perceptions of the MIS graduates.

The MIS faculty used data from the survey of employers and the survey of its graduates to develop a new curriculum for a B.S. in Management Information Systems.

BACKGROUND

The MIS curriculum is in a state of continuous change. The changing needs of students and the changing requirements of the MIS profession are two factors dictating a re-examination of the MIS curriculum.

During the 1988-89 academic year, the MIS faculty at Southern Illinois University at Edwardsville began work on the development of a new B.S. in Management Information Systems program. The existing program was a B.S.B.A. with a specialization in Management Information System consisting of seven courses. Increasing demands for highly-qualified MIS graduates influenced the MIS faculty to develop an MIS Bachelor's degree program.

Prior to designing the curriculum for the proposed B.S. degree in Management Information Systems, the MIS faculty decided to conduct a study of its graduates and employers to determine the competencies which were considered important for successful job performance.

OBJECTIVES

The purpose of the study of the attitudes of graduate and employers was to answer two questions:

1. What are the perceptions of recent MIS graduates regarding the skills and knowledge areas which are important for successful job performance?
2. What are the perceptions of major employers of MIS graduates with regard to the importance of these same skill and knowledge areas?

Based upon these data, the MIS faculty planned to develop a curriculum plan with objectives supporting the competencies stressed by alumni and employers.

METHOD

In the spring of 1989, a follow-up questionnaire was sent to 740 alumni who had completed the B.S.B.A. in Management Information Systems since the inception of this program in 1982. A total of 187 graduates returned the survey, representing a 25 percent response rate.

At this same time, the Department also conducted a survey of 170 employers of MIS graduates in the St. Louis and Southwestern Illinois metropolitan areas. The employers surveyed were members of the MIS Advisory Board and MIS executives listed in the Directory of Top Computer Executives. Thirty-four employers returned the questionnaire, representing a response rate of approximately 20 percent.

In addition to these data, in-depth discussions with members of the MIS Advisory Board helped the MIS faculty to develop new courses and to design the new B.S. curriculum.

RESULTS

This paper will present two sets of findings. The first set of findings describes how effectively current MIS offerings are preparing graduates for entry-level positions and the extent to which graduates feel that certain competencies are important for successful job performance. The second set of findings describes the employers' projected demand for entry-level MIS professionals and the skill and knowledge requirements for these positions.

Survey of Graduates

As indicated earlier, 187 graduates working in a variety of organizations responded to the questionnaire. Seventy-eight percent (78%) of the respondents indicated that the first job they obtained after graduation was in the MIS field, and all of these respondents felt that their undergraduate MIS program had been helpful in enabling them to obtain their first position.

These graduates worked in a wide range of organizations. The number of graduates working in firms within various size ranges is illustrated in Exhibit 1. As you can see, both large and small firms are represented:

EXHIBIT 1: NUMBER OF GRADUATES IN FIRMS OF VARIOUS SIZES

Size Range of Firm	% of Graduates
5000 or more employees	59%
1000 to 4999 employees	11%
Under 1000 employees	30%
Total:	100%

Graduates were asked to rate specific competencies taught in the program in terms of the importance to their current job responsibilities. Exhibit 2 below presents the results of the ratings. Respondents were asked to indicate whether each competency was very important (5), somewhat important (3), not important (1), or not applicable to their current work. The mean scores demonstrates that many graduates feel that communications skills, systems design skills, and database design skills are quite important to successful job performance.

EXHIBIT 2: GRADUATE RATINGS OF COMPETENCIES

Competency Area	Mean Score
Interpersonal skills	4.46
Speaking skills	4.23
Written communications	4.20
Basic systems analysis	3.93
Basic database design	3.47
Advanced structured systems analysis tools	3.28
COBOL programming	3.24
Job Control Language	3.16
Operating system concepts	3.09
Advanced COBOl	3.09
Basic data communications	3.05

Graduates also noted specific skills and knowledge areas which were important to successful performance in entry-level business computer positions.

These skills include on-line COBOL programming (CICS, or Customer Information Control Systems), systems programming in JCL (Job Control Language), and application programming in C language. Skill in database programming using mainframe database management systems languages, such as DLI and SQL, was also stressed by alumni responding to the study.

The importance given to interpersonal skills, speaking skills, and written communications provides support for the strong general education component of the proposed program. The importance given to the MIS competencies listed in Exhibit 2 provide support for the additional course work in data base management systems, communications systems, and systems design.

Survey of Employers:

Recent reports of the decline in demand for MIS-trained graduates influenced MIS faculty to ask employers about the anticipated demand for entry-level MIS professionals and to identify needed skill and knowledge requirements. As indicated earlier, about 20 percent of the 170 employers surveyed returned the Survey of Employers.

The objectives of the survey were to determine the number of MIS positions available for entry-level college graduates, to identify MIS positions of emerging importance, and to determine the importance of various competencies taught in the undergraduate program to successful job performance in entry-level MIS positions.

Graduates of the MIS program at SIUE have had considerable success finding challenging positions in the job market since the beginning of the program. The projected labor market demand for business-trained MIS graduates is very favorable. National and local studies continue to show that the computer field is a rapidly growing field of employment. Data collected from the follow-up study depicts a bright employment outlook for MIS graduates in the St. Louis area. The firms responding to the survey represent a variety of size ranges, as illustrated in Exhibit 3:

EXHIBIT 3: SIZE OF FIRMS

Size of Firm	% of Firms
5000 and over employees	30%
1000 to 4999 employees	43%
Under 1000 employees	27%
Total:	100%

Exhibit 4 shows the responses of employers pertaining to present and projected entry-level MIS positions. These data are based upon a 20 percent response rate.

EXHIBIT 4: EMPLOYER NEEDS FOR ENTRY-LEVEL MIS PROFESSIONALS

	Last Year	This Year	Next Year
Technical support programming			
Number of firms with any:	9	8	8
Avg. openings, if at least one opening exists	3.8	5.6	3.6
Programming analysis			
Number of firms with any:	19	14	21
Avg. openings, if at least one opening exists	3.8	4.4	3.9
End-User Computing			
Number of firms with any:	10	7	10
Avg. openings, if at least one opening exists	1.6	1.9	1.7
Data communications			
Number of firms with any:	4	4	5
Avg. openings, if at least one opening exists	1	1.25	1
Other openings			
Number of firms with any:	2	1	0
Avg. openings, if at least one opening exists	3.5	2	0

These data were used to forecast openings for the entire sample of 170 employers by multiplying the number of responses by four. A multiplier of four was used instead of five (the returns were approximately 20%) to be conservative. In this case, the projected number of firms with openings in programming/analysis for next year would be 84 organizations with an average of approximately four positions each. This would mean a total of 336 MIS positions for entry-level graduates in programming/analysis alone. In the next largest category of need, technical support programming, a projected total of 32 firms would have 4 positions each. This would represent approximately 128 programming positions. In the area of end-user computing, a projected 40 organizations would have an average of two positions each next year. This would represent an additional 80 positions in end-user computing. In data communications, an estimated 20 organizations would have one opening each, representing an additional 20 positions. In total, in all areas, there are 564 openings projected for entry-level MIS professionals in the population surveyed.

The projected number of positions represents a traditional market for MIS graduates. This traditional market includes large Fortune 500 and Fortune 200 firms with formal MIS organizations and MIS directors. A great many additional MIS positions exist in approximately 400 smaller firms in the St. Louis area ranging from $5 million to $200 million in sales. Educational institutions, consulting firms, hospitals, not-for-profit organizations and governmental agencies also represent employers of MIS graduates.

Positions of Emerging Importance

The MIS profession is rapidly changing as a result of new technologies, new organizational arrangements, and the emerging importance of end-user computing. Employers who were surveyed were asked to identify MIS positions of emerging importance within their respective firms. A summary of some of these results follows in Exhibit 5:

EXHIBIT 5: MIS POSITIONS OF EMERGING IMPORTANCE

Category of Position	Times Category Nominated
Communication/Networks	12
Management of MIS	11
System Development/Programming	11
Database/DB Administration	10
Microcomputer Support	5
End User Computing Support	4

Employees were asked to identify competencies which they considered important for successful performance in entry-level business computer positions within their respective organizations. The average ratings of these competencies are reported below (5 = very important; 1 = not important).

EXHIBIT 6: COMPETENCIES NEEDED FOR SUCCESSFUL JOB PERFORMANCE

	Average Rating
Interpersonal Skills	4.5
Written Communication	4.4
COBOL Programming	4.4
Basic Systems Analysis	4.3
Speaking Skills	4.2
JCL	3.9
Advanced COBOL (File structure)	3.8
Programming in 4GLs	3.6
Basic Database Design	3.2
Advanced Structured Systems Analysis Tools	3.1
Basic Data Communications	3.1
Operating Systems Concepts	2.9
Microcomputer Software	2.8
Advanced Database Design	2.7
Local Area Network Design and Management	2.7
Decision Support Systems	2.3
Expert Systems	2.2

As you can see from Exhibit 6, the most

important competencies identified by employers were effective communications skills, systems analysis, programming, and database design.

Comparative Analysis of the Attitudes of Graduates and Employers

The competencies rated most critical to successful job performance by graduates and by employers were remarkably similar. In Exhibit 7, you can see that both graduates and employers valued effective communications skills, basic systems analysis, and database design:

EXHIBIT 7: COMPARATIVE ANALYSIS OF GRADUATES AND EMPLOYERS

Competency:	Mean Score (Graduates)	Mean Score (Employers)
Interpersonal skills	4.5	4.5
Speaking skills	4.2	4.2
Written communications	4.2	4.4
Basic systems analysis	3.9	4.3
Basic database design	3.5	3.2
COBOL programming	3.2	4.4
Advanced COBOL	3.1	3.8
JCL	3.2	3.9
Operating system concepts	3.1	2.9
Basic data communications	3.1	3.1

As you can see from these data, the employers' perceptions of the importance of programming, both basic and advanced COBOL, was significantly higher than the perceptions of the graduates. A possible reason for this attitude may be that COBOL is still considered an important entry-level job qualification from the standpoint of employers. However, many MIS jobs today do not require extensive use of COBOL. Knowledge of JCL was also considered more important by employers than by graduates, probably because fewer graduates are involved in programming positions as compared with five to ten years ago. In narrative comments, however, some of the graduates stressed the importance of JCL, on- line programming, and database programming.

These findings are reinforced by other studies which stress the importance of interpersonal and communications skills in successful job performance (Albin and Otto, 1987; Hunter, 1987). A study by King also showed that oral and written communications skills were found to be essential in both the employer and graduate rankings.

Earlier studies have stressed the importance of COBOL to entry-level job performance, probably because most graduates assume positions as entry-level programmers and programmer/analysts. In contrast, this study seemed to indicate that programming was less important than systems analysis and database design at the entry-level. This may reflect the diminishing need for entry-level programmers and the increasing need for end-user computing analysts and systems analysts in an environment where diverse tools are being used to design and implement information systems.

Conclusions:

The survey of the attitudes of graduates and employers toward job requirements for entry-level MIS positions indicates a broadening of scope of skills and knowledge needed for successful performance. Rather than stressing programming skills at the entry-level, both graduates and employers are recognizing the importance of communications skills, systems analysis, and database design.

Job market data provided by the employer sample also indicated an increase in scope of entry-level positions. Although programming and programmer/analyst positions are still in demand, an increasing number of openings in end-user computing and in data communications are occurring. In the end-user computing area, in particular, interpersonal and systems analysis skills may be far grater in importance than technical programming skills.

Implications for the Curriculum

The follow-up studies of graduates and employers were used to design a proposed curriculum for a B.S. in Management Information Systems at Southern Illinois University at Edwardsville. The proposed curriculum consists of a general education core, a business core, and an MIS sequence composed of nine courses. The proposed sequence is shown in Exhibit 8:

EXHIBIT 8: PROPOSED MIS CURRICULUM

MIS 108 Introduction to Computers
MIS 342 Management Information Systems
MIS 260 COBOL I
MIS 270 Basic Structured Systems Analysis
MIS 360 Advanced COBOL
MIS 370 Advanced Structured Design
MIS 366 Database Design
MIS 466 Advanced Database Design
MIS 467 Communications Systems Design

One of the following:
MIS 462 Applied Operating Systems
MIS 472 Fourth Generation Tools

Required:
MIS 480 Information System Project

The basic difference between the former specialization in information systems and the proposed B.S. degree in MIS is the addition of advanced courses in structured systems analysis and database design. The need for these classes can be substantiated based upon the findings of the studies described in this paper. Another difference between the former program and the proposed program is that Basic Data Communications has become a required course. Because of increasing need for an understanding of communications systems design concepts in both production data processing and end-user computing positions, the MIS faculty believed that a basic data communications course was needed. Students have a choice from courses in operating systems and fourth generation tools.

In the future, as trends in the MIS profession require additional refocusing of the curriculum around areas such as end-user computing and telecommunications, additional follow-up data will be needed to fine-tune the curriculum. The studies reported in this paper were very valuable in establishing a need for the proposed B.S. program in Management Information Systems.

REFERENCES

Albin, Marvin and Robert Otto, "The CIS Curriculum: What Employers Want from CIS and General Business Majors," The Journal of Computer Information Systems, Summer, 1987, pp. 15-19.

Hunter, James, "What Topics Employers Think Should Be Included in CIS Courses," The Journal of Computer Information Systems, Spring, 1987, pp. 23- 26.

King, Albert S., "Revitalizing Management: Education Survey Design for Relevance in Business Schools," Mid-American Journal of Business, September, 1987, pp. 34-39.

A Data Communication Course for Information Systems and Computer Science

Donald G. Golden
James D. Schoeffler
Department of Computer and Information Science
Cleveland State University

ABSTRACT

Data communication courses are frequently found in both information systems and computer science curricula, with distinctly different approaches taken in the different curricula. That is, an information systems curriculum tends to focus on the uses of data communication and related management issues, while a computer science curriculum emphasizes the techniques used to implement these systems. This paper discusses a data communication course offered for both information systems and computer science students, which presents material relevant to both groups of students. In doing so, we present a paradigm that can be used to define the scope of courses in a combined information systems/computer science department.

INTRODUCTION

Our Department of Computer and Information Science has long offered a Bachelor of Science in Computer and Information Science as its undergraduate degree. Accordingly, the courses developed for the degree were designed for the needs of computer science students.

In 1986, in response to a growing need for students trained in the information systems field, the CIS Department began a second undergraduate degree, a Bachelor of Business Administration in Information Systems. Today, the graduates of both degree programs are computer professionals and have much in common. However, there are significant differences in their career paths and in the educational requirements for these paths. In particular, students in both degree programs need many of the same topics by name, but not in detail of coverage. Both programs are considered to be high level and it is important that information systems courses not be treated as "lower level" versions of the computer science courses, or that the information systems program be lacking to any significant degree in technical content. In other words, to educate students properly and prepare them for entry into the job market, both the information systems and computer science programs need to maintain high levels of quality and basic technical knowledge, in spite of the fact that students will be following different career paths.

The data communication area is one of the most important topics in current curricula because of the increasing trend toward distributed systems. In studying the data communication needs of both computer science and information system students, we find that there is sufficient overlap to allow a common data communication course. However, this requires that some topics traditionally taught in a data communication course be redistributed to other courses in both curricula.

Data communication topics in a computer science curriculum are usually oriented toward an understanding of the technical aspects of data communication hardware and software, such as the data transmission process itself (signals and modulation); organization of

communication interfaces and their support software; error checking schemes; organization of terminal emulation software, and so on. Most data communication textbooks directed toward computer science courses provide little application background as motivation, but rather dwell on the design and implementation of data communication systems.

In the contemporary information systems curriculum, data communication topics may be characterized in two forms: the "end-user" topics, and the "information system" topics.

End-user topics are those associated with user access of remote information. They include terminal emulation for using remote data and computer facilities; error-free transfer of data, particularly files; electronic mail; and other similar topics. Motivation for data communication frequently is provided by studying common end-user applications such as office automation and decision support. Often aspects of these applications are taught in information systems data communication courses in order to provide a basis for the discussion of data communication itself.

The information system professional is often associated with the specification, management of development, and operation of information systems for business organizations, and typically is interested in those aspects of data communication related to these functions. The topics include characteristics of wide and local area networks; transaction processing performance of the data communication networks; operation and management of networks; file transfer protocols; and the ISO multilevel network protocols. For this person, it is necessary to understand these and similar topics very thoroughly, but not to be an expert in the implementation of data communication software. In contrast, the computer science professional who may be carrying out the design and implementation of the information system from the specifications is very concerned with software implementation.

The needs of the information systems and the computer science curricula appear to be quite different when topics are listed as above. In fact, the similarity of needs is very much greater than is first apparent. Take, for example, the topic "terminal emulation." Both groups of students must understand the need for terminal emulation and the functions of terminal emulation. The latter is in the spirit of system specification, where the point of view taken is "what it does" as opposed to "how it does it." It is important to note that a specification implies a complete understanding of the system, not a superficial overview. It is true that a computer science student is concerned with the design and organization of the software required to implement a full-duplex terminal emulator. However, the current view of software engineering is that development starts with a specification, then proceeds to a design. For terminal emulation software, once the specification is understood the design process is not greatly different from other designs carried out in the systems programming area.

Generalizing from this example, we conclude that the division of data communication topics into separate specification and design areas permits us to select those topics which are common to both information systems and computer science (mainly the specification portion of the topics) and allocate them to a common data communication course, then assign the others (mainly software design and application motivation topics) to other more appropriate courses. In particular, we have moved most design-oriented aspects of data communication topics to system programming courses primarily in the computer science curriculum, and most application-

specific aspects to information system application courses primarily in the information systems curriculum.

THE ORIGINAL DATA COMMUNICATION COURSE

Originally, the data communication course for computer science students placed considerable emphasis on the implementation of communication software. Major topics in the course included physical modes of communication, asynchronous communication, terminal emulation, synchronous communication, and network protocols. Depending on the instructor, these topics could vary considerably in their presentation, but in general the course attempted to cover each topic from a discussion of its need through software implementation considerations.

The course typically began with a discussion of the physical characteristics of data transmission, then moved on to a consideration of asynchronous communication, including baud rate, parity, and error checking. A considerable amount of time was spent studying the RS232 interface and the roles played by the modem and UART; the UART in particular was studied at a fairly detailed level. Software design considerations included buffering I/O requests, use of interrupts, and concurrent access in a multiprogrammed environment. The material frequently involved a case study of the IBM PC, including low-level programming of an asynchronous interface with direct setting of interface parameters, and related programming assignments usually involved a study of terminal emulation in general.

After studying asynchronous communication, the course went on to look at synchronous communication and networks. Among the topics studied were error detection and recovery techniques, including CRC codes and their calculation; and link-level data communication protocols. Several examples of link-level protocols were studied, with special emphasis on BISYNC, DDCMP, and SDLC. The concluding topic, network protocols, studied both local area and wide area networks and their general use, beginning with the ISO multilayer network model. Emphasis was on software implementation issues and performance factors.

In summary, the course attempted to present the individual topics beginning with their roll in defining the needs and functional operation of data communication, and continuing through considerations of software implementation and performance. In practice, depending on the interests of individual instructors, much of the background material was ignored in favor of concentrating on software implementation techniques.

Topics in the data communication for computer science students were:

1. Physical modes of communication:
   - physical characteristics of data transmission media
   - frequency and amplitude modulation
   - frequency response characteristics of communication data lines
   - filtering, low pass (baseband), bandpass, and highpass systems

2. Asynchronous communication:
   - baud rate, parity, and error checking
   - the RS-232 interface
   - modem and UART (including internal block diagram operation)
   - character to bit stream conversion concepts
   - overrun, framing errors
   - software design including buffering of I/O requests, use of interrupts, concurrent access in a multiprogrammed environments
   - case study of IBM PC including detailed programming of an asynchronous interface including direct setting of interface parameters and a study of the programming of the

xxxx.

3. Terminal Emulation:
   The use of terminal emulation and the problems of supporting multiple keyboard/displays
   Software organization of terminal emulators

4. Synchronous communication :
   error detection and recovery including CRC codes
   Link level data communication protocols including their operation, characteristics of error recovery, and the software organization of their implementation
   Examples of link-level protocols special emphasis on BISYNC, DDCMP, and SDLC

5. Network protocols:
   LAN and WAN networks topology and use
   use of state diagrams for documenting protocols
   data compression techniques
   The ISO multilayer protocol including the objectives of each layer and its use in an application environment
   Software implementation issues in a multilayer protocol environment
   Performance issues in LAN and WAN networks.

Depending upon the instructor, these topics can vary considerably in their presentation. In general, it was common to attempt to cover each topic all the way from its need, its logical operation, and finally software implementation considerations. It was this breadth which we concluded was unsatisfactory to both CS and IS students, for different reasons.

THE REVISED DATA COMMUNICATION COURSE

As the objectives of the CIS Department evolved over time, it became clear that the data communication course was no longer serving the needs of our students. The majority of our students find jobs in the local area, an area whose industry emphasizes manufacturing and commerce rather than technology development. As a result, most graduates whose jobs involve data communication are not involved in the development of low-level software, but rather in installing and supporting communication systems using commercially available hardware and software. We concluded, therefore, that the attempted breadth of the course was unsatisfactory for both computer science and information systems students, for different reasons. That is, the computer science students frequently did not see the motivation and requirements specifications for the systems they were learning to implement, and the information systems students were being forced to study implementation details which were totally irrelevant to their interests and objectives.

The modified data communication course (and the consequent revision of other computer science and information systems courses) was based on the distinctions made between computer science and software engineering. That is, the overall data communication topic should be presented from a software engineering point of view, starting with system specification and proceeding to design and implementation, with the primary data communication course concentrating on the system specification and user requirements. For computer science students this provides the background and requirements specifications needed for software design, while information systems students become knowledgeable about data communication at a detailed level. This prepares information systems students for further courses concerning the problems of acquiring, operating, and managing large network systems.

In the spirit of teaching the specification as opposed to the design of a communication system, the topics of

the revised course are divided into user requirements, and communication system specification. Specifications range from pure "logical" (what is done but no mention of how it is done) through the "physical" (how the logical functions are carried out), but not to the software design level.

User requirements are concerned with the utility of the system as seen by the user. Topics include remote access of a computer for execution of application programs, the user's need for terminal emulation, transfer of files, the need for error detection and error recovery, and local and wide area networks and their user requirements.

Communication systems specifications are concerned with those details of the system which must be understood by both the information system specialist and computer science specialist, and include specification of communication media, the asynchronous RS-232 communication interface, synchronous communication concepts, network communication, and communication between computers; particular emphasis is placed on the ISO Open Systems Interconnection model.

Assignments in the course are directed toward two ends:

(1) improving the student's facility with the use of communication;
(2) improving the student's understanding and familiarity with the specification of communication systems.

For example, students use a microcomputer running a terminal emulation program, set communication parameters, then transfer files between a remote computer and the microcomputer using modems. In other assignments, students compare the efficiency of error detection algorithms using overly simple techniques such as vertical and horizontal parity, and good techniques such as CRC checking; or trace communication scenarios, including the response of the protocol, demonstrating understanding of the communication protocol by determining the response of the system during error recovery.

Textbooks for data communication are a problem. Too many assume that the information systems student needs only a superficial description of data communication. None that we are aware of makes the distinction between specification and design that we have proposed. Consequently, we have had to use a variety of sources of information in our course, including portions of textbooks on reserve in the library, copies of papers and articles, and material we have created. With another offering of the course, we believe the material will have solidified enough to begin writing a text directed toward this organization of a data communication course.

CONCLUSIONS

Our experience with the new approach (two offerings of the course) have been quite positive. The focus of the course on the specification of data communication systems defines clearly to both students and instructors the purpose of the course, the readings, the assignments, and lab experiences. This was not the case with the previous course, where the mixture of specification and design often slighted specification and design in favor of depth in system programming issues.

In developing the course, the new focus has helped immensely in planning projects and laboratory assignments. Moving topics to other courses has proved to be no problem (other than the extra effort required to coordinate courses). As a result, material that was formerly covered in the communication course has not been lost. For example, the study of implementing an interrupt-driven asynchronous communication system has simply been moved to a course on real-time systems. As a further result, we have had no

difficulty attracting both computer science and information systems students to the course, and both groups of students view the course as a useful part of their education.

Beyond the success of this one course, however, we feel that we have been able to identify a paradigm which can be used as a guide for organizing curricula in a combined computer science/information systems department. That is, the major emphasis of the information systems curriculum is the system life cycle through requirements and functional specification, combined with an introduction to managing the development and operation of computer-based systems. The computer science curriculum begins with the functional specification phase of the life cycle, but puts its major emphasis on design and implementation. We believe that this viewpoint will give us a valuable guide to organizing courses in order to avoid course duplication, while still serving both computer science and information systems students equally well.

BIBLIOGRAPHY

1. Campbell, Joe, C Programmer's Guide to Serial Communications, Howard W. Sams & Company, 1987.

2. Campbell, Joe, The RS-232 Solution, SYBEX, 1984.

3. Lane, Malcolm G., Data Communications Software Design, Boyd & Fraser, 1985.

4. Sherman, Ken, Data Communications, A User's Guide, Prentice Hall, 1990.

5. Stallings, William, Data and Computer Communications, Macmillan, 1985.

6. Stamper, David A., Business Data Communication, Benjamin Cummings, 1989.

PERSPECTIVES ON THE MASTER'S IN MIS

Jennifer L. Wagner
Roosevelt University

ABSTRACT

Because there are so few Master of Science in Information Systems (MSIS) programs, these programs are not universally understood. Their descriptions and curricula are far from standardized. This paper briefly describes two such MSIS programs and the rationale for their curricula. The perspectives and expectations of students, universities, and employers, which differ greatly, are then discussed. Finally, the implications of these varying perspectives on the MSIS program are examined.

INTRODUCTION

A relatively recent phenomenon in graduate business education is the appearance of the master's degree in management information systems. While the name of the program varies somewhat from institution to institution, it is often referred to as a Master of Science in Information Systems (MSIS). This term will be used in the following discussion.

The 1989 Directory of Management Information Systems Faculty in the United States and Canada (4) provides information on 445 bachelor's, master's, and doctoral programs in management information systems (MIS). Of these programs, only 48 (11%) consist of a complete MSIS program. The 1986 Directory (5) listed 47 MSIS programs. While this number seems amazingly static, the correspondence between the two lists is far from exact: 20 programs were listed in 1986, but not in 1989; 21 programs were newly listed in 1989; 27 programs were listed in both years. Because of the source of the data (self reports from questionnaires mailed to schools known to offer MIS) and the flexible response format (open-ended rather than multiple choice), inconsistency in the reporting, rather than appearance and disappearance of the programs themselves, may account for this discrepancy. It is, however, reasonably safe to assume that there are at least 27 MSIS programs which have been in operation for a minimum of five years. The schools providing these programs are listed in Table 1.

TABLE 1
Schools Offering MSIS Programs
in 1986 and 1989

Air Force Institute of Technology
Auburn University
Bentley College
Boston University
California State University, Sacramento
Claremont Graduate School
Clarkson University
Cleveland State University
Colorado State University
Creighton University
Eastern Michigan University
Georgia State University
George Washington University
Memphis State University
Naval Postgraduate School
Northern Illinois University
Rensselaer Polytechnic Institute
Roosevelt University
Southern Illinois University, Edwardsville
State University of New York, Binghampton
Texas A & M University
University of Arizona
University of Colorado, Denver
University of Miami

University of Missouri, St. Louis
Virginia Commonwealth University
Washington University

Because there are comparatively few such programs, curricula are far from standardized and the expectations of students, universities, and employers differ greatly. Following a brief description of two representative MSIS programs, the rationale for their stated curricula is examined. This paper then describes the perspectives, desires, and needs of the various consumer groups. It concludes with a discussion of the implications of these varying perspectives on the MSIS program.

TWO REPRESENTATIVE MSIS PROGRAMS

The first MSIS curriculum (7) described here consists of ten required courses. While the equivalent of an undergraduate business major is a prerequisite, no undergraduate computer-related courses are required. Programming is, however, strongly recommended. The seven graduate MIS courses are:

Introduction to Information Systems;
Systems Analysis;
Systems Design and Implementation;
File and Database Systems;
Accounting and Financial Information Systems;
Decision Support Systems; and
Management of Information Systems.

In addition, students complete three graduate business courses:

Production Management;
Statistical Inference; and
Managerial Accounting.

The second MSIS curriculum (6) consists of twelve courses. The equivalent of an undergraduate business major is a prerequisite, as is proficiency in COBOL (as demonstrated by an examination). The five required graduate MIS courses are:

Computer Concepts and System Software:
Systems Analysis;
Decision Support/Expert Systems;
Database Management Systems; and
Information System Design Project.

In addition, students complete three elective MIS courses selected from:

Advanced Software Support Systems;
EDP Auditing and System Controls;
Modeling and Simulation;
Decision-Making Heuristics and Expert Systems;
Management of the Information Resource;
Distributed Information Systems; and Research Problems.

Students also complete four graduate business courses:

Operations and Production Management;
Organizational Behavior in Management;
Environment of Business; and
Strategic Management.

RATIONALE FOR THE MSIS CURRICULUM

Both of these MSIS programs are based on a combination of professional and business curricular guidelines, as well as the demands of particular student bodies and educational settings. They incorporate some of the technical details of systems analysis, design, and implementation, while maintaining a conceptual and managerial perspective. The graduate course content is more theoretical than the content of corresponding undergraduate courses.

Although some courses may overlap, the MSIS is separate and distinct from the Master of Business Administration (MBA) with a concentration or specialization in MIS. The MBA is a more general degree and covers all of the functional areas of business, often including a course in information systems. Usually MBA programs require that students

specialize in one of these particular functional areas, which may, at some schools, include MIS. Thus, the student pursuing the MBA with a concentration in MIS would complete approximately 20% of his course work in MIS.

The MSIS, which, like the MBA, resides in the business school, is a more specific degree program. The majority (about 70%) of the courses required toward the MSIS are in MIS; the remainder of the program consists of some general business courses.

Although some may disagree (as described below), a recognized purpose of the MSIS program is to prepare the student to manage MIS projects and/or personnel. Many authors (2, 8) agree that managers require education in three areas: conceptual skills, technical skills, and human skills.

Conceptual skills: A primary goal of graduate education is the development of a research orientation so that students can keep up with the field after the conclusion of their formal coursework. Because MIS, however, tends to be technology-based (if not technology-driven), maintaining an appropriate conceptual level, without becoming embroiled in technological detail, is a challenge faced by the MSIS program curriculum developers. It may well provide the basis of much of the variation among the consumer perspectives.

Technical skills: In order to function effectively, an MIS manager must have sufficient knowledge of the technology to be able to communicate effectively with vendors and the MIS staff. Thus the MSIS program provides the essential technological concepts. Graduate courses include some basic technical detail while emphasizing the conceptual basis and overall framework underlying the technology. While undergraduate students learn how to use various MIS tools and techniques, graduate students learn <u>why</u> as well as <u>how</u>.

Human skills: Managerial issues, such as dealing with employees, budgeting, scheduling, planning, and communicating are included.

STUDENT PERSPECTIVES ON THE MSIS

Students want the MSIS program to provide the knowledge and credentials which they need in order to move up the corporate ladder, with increasing salary and responsibilities. Some students wish to become effective managers in the information systems field itself; others wish to utilize the information resource more effectively and efficiently within their current functional areas. Thus the knowledge which the students think that they need (and the knowledge which they do, in fact, need) varies from student to student.

The MSIS program serves two major groups of graduate students: (1) those with computer-related undergraduate degrees, who understand that they must increase their knowledge of business and managerial topics; and (2) those with business or other undergraduate degrees, who perceive that they must broaden their knowledge of the somewhat-technical information systems topics. Members of both groups may wish to follow career paths leading eventually to the position of chief information officer (CIO) of the organization.

While those students with prior computer-related expertise often expect to move along the traditional programmer - analyst - manager of systems development - CIO path, students who have had little computer-related education may expect to follow a different path. Their functional area experience will prepare them for positions such as user liaison and information resource manager. For example, a student with an undergraduate major in accounting may pursue the MSIS in order to become a user liaison or "translator" between the user-accountants and the system develop-

ers. A student with an undergraduate major in library science may see a career as a database administrator after he acquires the MSIS, while a major in management and an MSIS degree may lead a student to seek the position of information resource manager.

In addition to their personal motivations, graduate students bring a variety of undergraduate majors and industry employment into the MSIS program. These prior experiences further color their expectations and needs. Frequently, these students have made immutable their choices of career, industry, and even particular organization. Many graduate students wish to pursue their education on a part-time basis while continuing full-time employment. Such students often demand that course material be immediately applicable to their present workday situations and problems.

UNIVERSITY PERSPECTIVES ON THE MSIS

It is virtually impossible, except with anecdotal evidence, to separate faculty perspectives from those of the university administration. These two viewpoints will thus be considered together here. A university which offers a doctoral program in MIS may incur few, if any, additional costs by offering the MSIS. However, a school which does not already have MIS technology and faculty on hand may encounter some difficulties in offering the MSIS. The 1989 Directory (4) identifies 29 MSIS programs at institutions which do not also offer a doctorate in MIS. It is unlikely that their rationales for offering such programs can be precisely determined. However, three reasonable assumptions are that such programs enhance the reputation of the school, that there is student demand, and that the graduates will become successful managers. There are two overlapping factors which inhibit the success of an MSIS program: the rapidly changing computer technology and the lack of qualified and available faculty.

Rapidly changing technology results in hardware and software acquisition costs, which may overwhelm the university budget. However, this difficulty can be overcome. Faculty expect to teach theory, rather than technology, to graduate students; this should serve to limit the demands placed on the computer facilities. Furthermore, many part-time students use new technologies in their workplaces; they may be able to share their experiences with other students and faculty through class discussions or on-site demonstrations.

The availability of qualified faculty presents a greater problem. The wide range of MIS activities, involving, as they do, people, computer hardware and software, and raw data, requires an equally wide range of managerial skills and knowledge. It is unlikely that such diverse skills and knowledge can be readily acquired by non-MIS specialists without extensive formal training. Even faculty holding a doctorate in one area, say, computer science, may have extreme difficulty picking up the needed skills and knowledge in the other areas, say, basic business functions. Thus the faculty who teach in the MSIS program must have either earned a doctorate in MIS or acquired some other formal training in MIS.

TABLE 2. Percent of MSIS Faculty with MIS Doctorate

Percent of MSIS Faculty with MIS Doctorate	Number of Schools
0	9
1 - 19 %	7
20 - 39 %	6
40 - 59 %	5
at least 60 %	2
TOTAL	29

As shown in Table 2, most of the 29 MSIS programs at institutions which do not also offer an MIS doctorate are staffed by faculty who do not hold a doctorate in MIS. Of course, these faculty members are quite likely to have extensive industry experience or additional formal education in MIS. However, the lack of MIS doctorates is of concern to the accrediting agencies and may affect the overall reputation of the university.

EMPLOYER PERSPECTIVES ON THE MSIS

Employer perspectives are by far the most difficult to assess. However, they are also the most interesting. Recent articles in the MIS professional press emphasize the need for business training of MIS staff. "The growing trend has been to hire and promote those who not only have solid technical foundations, but who possess an understanding of the company, the business and the applications" (1, p. 36). "[I]nformation managers need to improve their critical thinking, communication skills, and ability to incorporate new technology into current systems" (3, p. 60).

Such comments stressing the need for graduate business education usually discuss the MBA with a specialization in MIS. While such education may be adequate, it seems very possible that prospective employers would be eager for MSIS graduates. Until the employers are aware of the MSIS and its advantages, they are unlikely to seek out MSIS graduates.

CONCLUSION

The perspectives of the various audiences for the MSIS program differ. Some MSIS curricula are considerably more technology-based than others. Some students see the MSIS as a very managerial business degree; other students see it as a highly technical computer-based program. Universities and faculty perceive a program which places substantial demands on limited resources, but maintains contact with tomorrow's high-level managers. Employers and other MIS professionals seem to be unaware of the MSIS. Before the MSIS becomes a widely accepted degree, these various viewpoints will have to merge.

REFERENCES

1. Israel, Burt, "Hiring Systems Analysts for the 1990s", Information Executive, Winter 1990, pp. 35-36.

2. Katz, D., "Skills of an Effective Administrator," Harvard Business Review, volume 52, 1974, pp. 90-102.

3. Krass, Peter, "The MBA Influence on MIS", Information Week, January 29, 1990. pp. 56-61.

4. MISRC/McGraw-Hill, 1989 Directory of Management Information Systems Faculty in the United States and Canada, McGraw-Hill, 1989.

5. MISRC/McGraw-Hill, 1986 Directory of Management Information Systems Faculty in the United States and Canada, McGraw-Hill, 1986.

6. Pollack, Thomas A., "The Master of Science in MIS: A Model Curriculum which Complies with AACSB Standards", ISECON '88 Proceedings of the Conference Sessions, Data Processing Management Association, 1988, pp. 227-230.

7. Wagner, Jennifer L., "Curricular Differences Between Graduate and Undergraduate Programs in Information Systems", ISECON '88 Proceedings of the Conference Sessions, Data Processing Management Association, 1988, pp. 223-226.

8. Wagner, Jennifer L. and Sally A. Dresdow, "The Role of the CIO", 1987 HRMOB Proceedings, Volume II, The Association of Human Resources Management and Organizational Behavior, 1987, pp. 227-231.

WHY, WHAT, AND HOW TO TEACH THE DATA STRUCTURES COURSE TO CIS STUDENTS

Hassan Pournaghshband
Department of Math/Computer Science
University of Missouri-St. Louis

Alireza Salehnia
Department of Computer Science
South Dakota State University

ABSTRACT

The purpose of this paper is to help the instructors, particularly those teaching in Computer Information Systems programs, to improve the quality of their data structures classes. Some of the frequent problems encountered while teaching this course are examined and suggestions for minimizing these problems are given. Based on the authors' experiences in teaching the data structures course to computer science and computer information systems students, it is shown "what" to cover and "how" to teach this course for CIS students.

INTRODUCTION

The computing trends in the 90's are toward business applications and user orientation programs. Many courses which were taught only in computer science (CS) departments, now are also taught in computer information systems (CIS) departments. Many large and small educational institutions are teaching CIS courses and offer CIS programs. While "computer scientist" is placed at the most technical end, "information architecture" is placed at the end emphasizing organizational concerns [3].

The precise role of data structures, which is very clear for CS the curriculum is yet to be clarified for CIS program. This is due to the fact that there are differences between the entering students and graduates of CS and CIS programs. Students entering the CIS program come from a variety of backgrounds. They may come from engineering and science or from business disciplines. And once they graduate, there are a variety of positions in the market that fit their qualifications. Some of these positions are highly technical while others are less technical and more organizational. The authors have taught the data structures course to both computer science and computer information systems students, and have experienced the effects of these differences in both environments.

In this paper, we do not intend to investigate the two curriculums, since this has been intensively done by the ACM and the DPMA curriculum committees [2, 4, and 7]. The main purpose of this paper is to concentrate on the data structures course and to examine those problems involved in teaching this course to different audiences from different backgrounds. We discuss these problems for the CIS program since we believe as stated earlier, that this will be one of the most rapidly growing fields for the 90's. We will share our experiences, and we will give recommendations toward improving the quality of the course, thus helping the instructors to utilize the course more effectively.

DISCUSSION

In 1979, the ACM curriculum committee on computer science proposed recommendations for the undergraduate curriculum in computer science: "curriculum '78" [4]. This curriculum includes CS7, "Data Structures and Algorithm Analysis" as a core requirement. Later in 1982, the ACM curriculum committee on information systems proposed recommendations for the undergraduate and graduate programs in information systems for the 80's [7]. This one does not include a data structures course similar to CS7, but includes IS2, "Program, Data, and File Structures" as a core requirement. A recent study by Chen and Willhardt [1] indicates that only 36 percent of the AACSB accredited business schools offer IS2 in their CIS programs. The most recent DPMA recommendation for baccalaureate degree programs, "CIS '86", does not even recommend data structures as a required or an elective course [2]. DPMA recommends two business application programming courses, namely CIS/86-3 "Introduction to Business Application Programming" and CIS/86-4 "Intermediate Business Application Programming". These two courses are oriented toward COBOL and other high level programming languages. COBOL programmers need not to be concerned with the detailed methods in which data structures are programmed. However, it is important for them to understand how different data structures function and the application for which the data structures are implemented.

In the following sections we will examine a data structures course for the CIS programs and will discuss "why", "what", and "how" to cover this course for CIS students.

RATIONALE

An important characteristic of the data structures course is its problem-solving nature. Users and programmers must understand the underlying data structures of an application program in order to be able to utilize, repair, or modify it efficiently. Therefore, this can be a very interesting course and an enjoyable experience for the students if they "understand" the material and if they follow the course throughout the semester. On the other hand, the course can easily become a nightmare for them if the topics are covered vaguely and the relationships between different structures and real world problems are not clarified. One effective solution to this problem is to teach and encourage students to apply their knowledge of data structures. This can be done by (1) designing data structures for given applications, (2) assessing the quality of a particular data structure within a given context, and/or (3) designing, comparing, and selecting alternative data structures or algorithms for a given situation.[6]

DATA STRUCTURES FOR COMPUTER INFORMATION SYSTEMS

As mentioned earlier, ACM Information Systems Curriculum recommendations for the 80's does not include a data structures course similar to CS7. It however, includes IS2, "Program, Data, and File Structures" which includes, among other topics a unit on data structures and indexing. While the DPMA and ACM curriculum guidelines support schools for developing or redesigning their curricula, not many institutions offer a separate data structures course for their CIS program [1]. The authors have taught advanced CIS courses in both types of schools, i.e., those with a data structures course and those without it. Based on our experiences, we strongly believe that data structures should be a central course (or main topic of other related courses) for information systems programs as it is for CS. However, "what" to cover and "how" to cover it should differ from those of the CS programs.

To discuss "what" to cover, we first suggest the objectives of the course and then discuss the specific topics. The course objectives should include:

A. To help students learn what data structures are, how to manipulate them, and when to use a particular data structure.
B. To introduce concepts and techniques of structuring data on storage devices.
C. To provide students with an opportunity to design and implement data structures.
D. To provide students with an opportunity to understand and solve "real problems".
E. To provide students with an opportunity to analyze alternative solutions for a given application, and to select the most effective data structure for it [6].

In general, a major concern should be the manner in which data is stored, retained, controlled, and accessed for processing. It should be emphasized how a proper choice (e.g., sequential access vs. random access) of a structure for the data in a particular application can significantly simplify the programmer's job and enhance the processing performance.

We start the course with a review of file structures. Different access methods and their applications are discussed. We always include in our outline, an example to show how a bad choice of data structure significantly slows the processing down. Next we cover arrays and linked lists. We carefully examine these two structures and compare their applications. Our students appreciate using linked lists when an application which requires many additions and deletions is given. Stacks and queues are covered next. Both array representation and linked representation of these structures are discussed. At this point, we cover sorting and searching (both internal and external) in detail. For each sorting/searching method we analyze the time and space complexities, but we do not get involved with formal proofs. Finally, the tree structures is discussed in detail and nonlinear and linear structures are compared. Binary trees, decision trees, and B-trees are discussed in more detail.

We recommend the following topics for the course.

I. INTRODUCTION TO DATA STRUCTURES. Basic terminology and programming conventions are covered, and the rationale of the course for the CIS program is explained.

II. LISTS. Arrays and linked lists are discussed in detail. The criteria for choosing one structure over the other one are also examined.

III. RESTRICTED DATA STRUCTURES. Stacks and queues are covered. Examples, using different structures for the same application, and different structures for different applications, are given.

IV. SORTING AND SEARCHING. Different techniques for internal and external sorting/searching are discussed and their performance for different situations are examined.

V. TREES. Binary trees, Multiway trees, Decision trees, and B-trees are discussed.

We recommend modifications and additions such as Decision Tables, Graphics, and Networks to the above list when necessary. The modifications should depend on the needs of the educational institutions and the community they serve. Instructors should select essential topics which satisfy these needs.

With regards to "how" one might teach the course, our experiences have shown

us that it can be (and should be) taught without the use of formal disciplines, and in a manner that is familiar to students in a business-oriented environment. Our experiences shown us that one frequent problem in teaching data structures to CIS students is having them enrolled in a single class, while their backgrounds range from those having several upper level division courses along with a significant amount of work experience to those having only completed one or two introductory course(s) with no work experience. For this class, the programming side of data structures must be emphasized, and a detailed discussion of analysis of algorithms should be avoided, except possibly for sorting and searching methods. Real world examples significantly seem to enhance the quality of the course and are appreciated by most of the students. To better encourage our students, we always began with simple examples that did not require complex data structures, and more complex problems were used later in the semester after several data structures had been covered, and students became comfortable using them.

In order to effectively achieve the objectives of the course, each instructor may use his/her method according to his/her outline and other criteria. One method which we have effectively used in our classes is discussed below.

At the beginning of the semester, a "real" project such as "airline reservation" was assigned. The students were asked to design and implement a solution to the problem. Throughout the semester, while covering different data structures, the instructor would discuss these solutions and their flaws, and suggest alternative solutions. The same project was assigned as the final project. This time, students were asked to solve the problem as efficiently as possible, using their knowledge of data structures. The students were also asked to compare the new solution with the one given at the beginning of the semester.

Obviously, an appropriate textbook for the course can greatly help the students (and the instructor as well) to get a better outcome from the course. We recommend a textbook such as [5].

SUMMARY

The authors' experiences in teaching the data structures course, particularly to students in a Computer Information Systems program, were discussed. We examined frequent problems encountered in teaching this course and gave suggestions for improving the quality of the course.

REFERENCES

1. Chen, J. and Willhardt, J. (1988). "Computer Curricula in AACSB Accredited Business Schools." Interface, 10(1) (Spring 1988), 28-31.

2. "CIS '86: The DPMA model curriculum for undergraduate Computer Information Systems." Interface, 8(3), 33-65.

3. Cotterman, W. (1983). "A Comparative Analysis of Information Systems Curricula." Proceeding of the Second Annual Information System Education Conference, Chicago, 1983.

4. "Curriculum '78: Recommendations for the undergraduate program in computer science." Communications of the ACM, 22(3), March 1979, 147-160.

5. Ellzey, R. (1983). Data Structures for Computer Information Systems. Chicago, IL: Science Research Associates.

6. Hartman, J. and Chrisman, C. (1987). "Providing activities for students to apply data structures concepts." ACM SIGCSE Bulletin, 19(1), 336-340.

7. "Information systems curriculum recommendations for the 80's,: A report of the ACM curriculum committee on IS." Communications of the ACM, 25(11), November, 1982. 781-805.

Teaching Computer Ethics

Carol S. Chapman
Haywood Community College

ABSTRACT

The morals of computer ethics must be taught throughout a computer curriculum if graduates are expected to engage in ethical behavior in the work place and in their lives. The content of information that should be covered and what structure and courses(s) should take are often in question. The many ethical and unethical behaviors should be covered in such a structure that involves and intrigues the student.

A Strategic Approach to Teaching Communication Skills to CIS Students

Diane Fischer
Carol Okolica
Dowling College

ABSTRACT

This paper presents a Computer Information Systems course in communication and reporting against the background of the need for improved writing skills of undergraduate students. It addresses the needs of students in general, and discusses the academic responses of remedial writing courses, core curricula and writing across the curriculum. CIS students have, by virtue of their discipline alone, writing needs that go beyond these general responses. A CIS course in communication and reporting addresses these needs. A syllabus for such a course is presented, along with discussion of how this course fits the needs of CIS students at a small liberal arts college.

Internship - A New Approach

Christine B. Kay
DeVry Institute of Technology, Chicago

ABSTRACT

This paper describes a new approach to internship programs in which students in a senior computer course spend the length of the course working on a "real world" computer project in a real company. This paper also describes the forms and procedures used for implementing this kind of internship program with regard to the school, students, teacher and companies.

ETHICAL ISSUES IN COMPUTING - STUDENT RESPONSES

Riva Wenig Bickel*
Department of Computer Science
State University of New York, College at Oswego, Oswego, NY 13126 (315) 341-2367

Maria M. Larrondo-Petrie**
Department of Computer Science, College of Engineering, maria@cs.fau.edu
Florida Atlantic University, Boca Raton, FL 33431 (407) 367-3855

David Bush
Department of Psychology and Graduate Program in Human Resources
Villanova University, Villanova, PA 19085 (215) 645-4746

Fred G. Harold
Department of Computer Information Systems & Decision Sciences, College of Business
Florida Atlantic University, Boca Raton, FL 33431 (407) 367-3181

ABSTRACT

An instrument named EDICT (Ethical Dilemmas In Computing Test) is being developed to collect and analyze responses to eight dilemmas encompassing multiple aspects of computer-related ethics. Details of a preliminary study to revise and improve these scenarios are reported, and selected student responses are discussed as they relate to the authors' ongoing efforts to provide educational methodology to address ethical issues associated with computer use.

INTRODUCTION

While the computerization of our society brings us enormous benefits it also exposes us to numerous threats. We are vulnerable both financially and physically to those who possess a high degree of computer expertise but no corresponding high ethical standards as constraints upon their activities.

Although computer crime is only one area where unethical behavior vis a vis computers impacts society, it elicits the strongest visceral response. Consequently, there is a general call for tighter security, stiffer criminal penalties and an increased willingness to prosecute computer criminals.[1,2,3,4,5,6] Such measures are necessary, and offer partial protection, but alone are inadequate.

Computer criminals, once categorizable primarily as bright young males[7], now include both sexes and all ages. Many computer criminals still see their acti-

* This research was done while the author was at Florida Atlantic University and was supported in part by an internal seed grant from that institution.

** This author was supported by a McKnight Foundation Fellowship through the Florida Endowment Fund.

vities as swashbuckling adventures rather than criminal acts. They are intelligent enough to realize that they have little chance of being caught; very often such crimes would not have been discovered at all but for misfortune or the activities of an informant.[8,9] There is little reason for them to fear the severity of possible penalties--increased levels of security present additional challenges while inconveniencing legitimate users.

EDUCATION AND ETHICAL COMPUTER USE

Educators, psychologists, criminologists and computer professionals all agree that education in ethical computer use is an important avenue for attacking computer crime.[10] Other important ethical issues include social equity, workplace disruption, privacy protection and responsibility for defective or inadequate products.

One of the authors (Bickel) presented students in four classes with a number of computer-related ethical issues using previously published scenarios.[11,12] While most students, at least publicly, did not condone trespass to other people's computer databases, many found no problem (and some even merit!) in copying computer software. Most did not consider invasion of privacy an ethical issue, and a majority would create any program their bosses required, even to serve illegal purposes. Classroom discussion helped sensitize the majority of students to these issues; for some, however, the attitude of "getting away with whatever you can" was deeply ingrained, persisting even in their responses to exam questions concerning ethics. This may indicate the students were not even aware that certain acts were unethical. Clearly, there is a need for educators to instill in their students a sense of ethics and mature responsibility for their activities.

THE RESEARCH PLAN

With this need in mind we began a multiphased research plan to create an instrument, the Ethical Dilemmas in Computing Test (EDICT), designed to elicit information that can be used to develop effective methodology for teaching computer ethics at each instructional level.

In phase one we created scenarios together with questionnaires that could be utilized on a large number and variety of people: computer professionals, business people, and university and secondary school students. This paper presents some preliminary results based on an in-depth investigation of responses to 8 scenarios by 18 university students.

In phase two, the format of the final instrument will be modelled after Rest's Defining Issues Test (DIT).[13] The DIT has been extensively used to ascertain stages of moral development and has proven useful in a number of applications, including the evaluation of over fifty studies to determine whether moral education improves moral judgment.[14,15]

During phase three, submission of the DIT together with the EDICT will allow correlation of results and validation of the EDICT. Subjects will be asked to react to scenarios by: (1) deciding whether or not to take a given course of action, (2) scoring on a scale of 1 to 5 the significance each of 12 given considerations had on their decision, and (3) ranking the 4 most important considerations contributing to their final decision.

The format of this instrument is convenient for both subjects and researchers. The subject responds simply by filling in a few blanks for each scenario; the standardized answers can be scanned and evaluated with the aid of a computer.

CURRENT RESEARCH DESIGN

The 8 EDICT scenarios were designed to present realistic dilemmas in that the situations are not clear cut; there are strong justifications supporting each alternative course of action. A brief summary of the scenario topics follow:

(1) Proprietary Software. Should an employee, facing a layoff due to a hostile

takeover, use proprietary software he helped develop to enhance his marketability in finding a new job?

(2) Credit Database. Should a computer expert comply with a friend's request to look in her computer credit files verify its accuracy?

(3) Medical Database. Should the owner of a company that keeps a medical database to assist clients in filling out medical forms sell information to a pharmaceutical company for promoting new products?

(4) Educational Software. Should a teacher in an impoverished inner-city school accept a donation of computers and violent, inferior software in exchange for using and promoting only the vendor's software?

(5) Job Displacement. Should a system analyst reveal a highly original plan which will make the department extraordinarily efficient but cost even more jobs than anticipated?

(6) Unused Patents. Should an employee "blow the whistle" when research based on some patents beneficial to the public is halted by the company so it can use its resources more profitably?

(7) Student Access. Should a student programmer who has depleted his computer time attempt to transfer excess time from accounts of classmates?

(8) Instructional Copying. Should a community college instructor make unauthorized copies of popular software packages so her class can learn to use them?

The motivation for our preliminary study was to gain a full understanding of students' reaction to the scenarios, and identify factors which influenced their responses. Omissions, ambiguities, misleading statements in the scenarios were revealed when the subjects were not constrained to choose from predetermined responses, as they will be with the final version of the EDICT.

The experiment, designed to validate future scenario responses, included a detailed study using 18 undergraduate students in a Computers and the Law class. First, the students were asked to respond in essay format to 8 scenarios, stating (1) what advice they would offer the hypothetical person, (2) what considerations went into that advice, and (3) what additional information would have made them feel more comfortable with their recommendation.

The second step, a class discussion of the scenarios and responses, permitted students to hear other viewpoints and alerted them to issues which they might not have previously considered. The researchers' role in this discussion was to elicit the participation of all class members and encourage a depth and breadth of discussion while remaining non-judgmental to encourage free expression. In addition to moderating and recording the discussion, the researchers served as information resources, especially with respect to the legal aspects of some of the scenario situations. Opinions on the various issues were often strongly held and at times led to vigorous debate.

In the third step the students rethought their responses to the scenarios in view of the class discussion. They indicated whether they had changed their opinions and if any part of the discussion had influenced their final answer. They also ranked, in order of importance, those considerations which had gone into their final answer.

RESULTS

The results were not only inherently interesting, but also indicated a number of weaknesses in the scenarios. For example, consider the revised Educational Software scenario:

"Sarah Frost teaches sixth grade students in an inner city school and is in charge of acquisitions. She is approached by a software company, Ed's Software, with a proposal. They will provide for her

class two free computers and a selection of their educational software games if she in turn, will promise to use only their software on the computers and, also to allow other educators, who are making software purchasing decisions, to observe her class using the computers. The implication is quite strong that she will be expected to recommend the games to these visitors. The school administration does not know about this proposed deal.

"Sarah goes to Ed's Software and views their programs. She notes that the educational games that she is being offered do cover the sixth grade curriculum in math, but that the games are all very violent. There is a lot of educational material in the market that presents the same material in a non-violent way. The English skill games do not appear to her to be very good. However her school cannot at this time afford to purchase any computers or software and if she does not accept the students will have no computers and no software.

"Should Sarah accept the donation from Ed's Software?"

Issues of interest arising from this scenario include the paucity of computer education opportunities for disadvantaged students and the exposure of school age children to violence. However, in the original scenario the students were third graders; third graders, according to many of the subjects, do not urgently need computer skills. The original scenario also positioned Sarah as a member of a computer acquisitions committee rather than the sole decision maker; subjects could thus spread the accountability to other committee members rather than being forced to reach a decision on their own. On the average there was one change per scenario necessary to either remove an ambiguity or to prevent the subject from finding an easy answer to what was supposed to be a difficult dilemma.

We were particularly interested in learning what factors the students felt might sway their decisions. Prior to actually performing the experiment and in conjunction with developing the scenarios, a preliminary draft of a selection of predetermined responses was composed; these needed to be validated and the most salient ones chosen for the final version of the EDICT. Many issues considered for the predetermined response selection were raised independently by the students. Examples from the Educational Software scenario included:

(1) Can Sarah, using Ed's offer, negotiate with a better software developer?

(2) Are there too many students in the class to profit from 2 computers?

(3) Will violence make the packages more attractive to the students?

(4) Would outside observers distract the classroom, thus negating any benefit?

(5) What is the level of violence employed--shooting aliens with ray-guns or attacking humans with chainsaws?

(6) How will parents react to any unilateral decision by Sarah?

(7) Can Sarah use the bare machines to teach the students programming?

(8) How will accepting the donation affect Sarah's job and her relationship with her employers?

Because one intended use of the scenarios is to determine the moral development stages indicated by the responses, an attempt was made to associate each unstructured answer with one of Kohlberg's six moral development stages: [16]

Stage 1. Obedience and punishment: Acts to stay out of trouble.

Stage 2. Naively egotistic: Acts expecting reciprocity.

Stage 3. Good boy/good girl: Desires to please and obtain approval.

Stage 4. Social system/legal authority: Acts in support of society and laws.

Stage 5. Contractual legalistic: Motivated by utilitarian principles.

Stage 6. Universal principles: Abides by universal ethical principles (e.g.justice)

One result was compelling: Although a fear of adverse repercussions is considered Stage 1 (the lowest stage) of moral development, each one of the 18 students responding mentioned negative consequences at least once in their essays. This might be due to the form of the question; subjects were asked to advise a person facing the dilemma, rather than state how they themselves would handle it. In giving advice to others it is natural to mention the possible consequences -- especially when those might involve imprisonment or financial penalties. Encouragingly, in only 2 or 3 instances was the Stage 1 consideration a primary one.

Loyalty, a stage 3 manifestation, was mentioned frequently. An interesting variant was employee loyalty -- loyalty to an entity rather than an individual. In the scenarios 1, 5 and 6 (where employee loyalty could be a critical factor) 10 of the 18 respondents specifically mention loyalty to the company in at least one of the scenarios. However, all 3 subjects who also gave rather cynical answers to some of the scenarios came from this group. An example of a cynical answer from the Educational Software scenario: "if she does not have to sign a contract ... and the goods are over $500, then I would take the goods and use any software I wanted." The latter refers to the common law requirement that sales contracts for over $500 be in writing to be valid; however, there was no sale involved. On the other hand, of the 7 students who volunteered higher stage ethical statements, only 3 were also in the group mentioning loyalty to the employer.

One of the most encouraging results was the finding that class discussion often changed people's chosen action -- or at least sensitized them to factors previously unconsidered (e.g., employee retraining). There was an average of 4 changes per scenario, representing 23% of the replies. Only 4 subjects (22% of the class) were not induced to change a single answer. In some cases, changes were due to a new-found awareness of applicable laws. For example, once students realized that the Fair Credit Reporting Act protected consumers who are improperly denied credit, they were less likely to condone self help via hacking into credit agency databanks. Knowledge of legal loopholes, however, could prove antithetical to ethical behavior; one subject, learning that state institutions can break the copyright law with impunity (relying on sovereign immunity), changed his response from "don't copy copyrighted software" to "do copy it".

Two scenarios evoked passionate personal statements -- one from a faculty member observing the process and one from a student. The latter was most indignant in responding to the Instructional Copying scenario. "I know everyone does it," she said "and I used to do it myself. But now I work as a programmer and it is hard work. I would really resent that someone used my work without paying for it." (This statement involved the concept of fairness rather than profit -- she was clearly an employee rather than an entrepreneur). In both cases, students responding to the third part of the assignment cited these specific people as having strongly affected them. Yet neither impassioned orator changed an extraordinary number of minds; in fact one student who held his ground stated specifically that he did so despite the faculty member's contrary opinion.

CONCLUSIONS

These are some of the observations made on responses to the scenarios. Given the nature of our daily headlines it was encouraging to see how many students gave thoughtful, well-reasoned, and ethical answers to tough dilemmas, and, also, how open-minded most students were. Although

the sample size is too small to be of statistical significance, we were able to improve our scenarios prior to use on a larger set of subjects. The task of validating the questionnaires and determining the effects of previous courses in ethics and/or computer sophistication is now under way.[17] The authors hope these results, and those obtained from future phases of this research, will provide guidance to those interested in ethical education.

ACKNOWLEDGEMENTS

The authors wish to thank for their able assistance: Eric Share, Michael Copenhaver, Mary Stevens, Yonith Bickel, and the Computers and the Law students at Florida Atlantic University.

REFERENCES

1. Florida Statutes, Ch. 815, Computer-Related Crimes (1985).

2. H.W. Cates, "Computer Crime Law: Review of State Statutes", Data Processing and Communications Security, 10, 2, Spring 1986, pp. 19-21.

3. Computer Security Act of 1986, 99th Cong. H.R. Rept. 2889.

4. Computer Fraud and Abuse Act of 1986, 99th Cong.

5. C. Stoll, "Stalking the Wiley Hacker", Communications of the ACM, 31, 5, May 1988, pp. 484-497.

6. People v. Versaggi, 518 N.Y.S.2d 553 (City Ct. 1987).

7. S.R. Wolk, W.J. Luddy Jr., Legal Aspects of Computer Use, Prentice-Hall, 1986, pp. 121-122.

8. U.S. v. Seidlitz, 589 F.2d. 152 (4th Circ 1978).

9. M. Hochman, "The Flagler Dog Track Case", Computer Law Journal, Summer 1986, pp. 117-126.

10. M. Alexander, "Strong scruples can curb computer crime", Computerworld, 3 April 1989.

11. E.A. Weiss, ed. from book by D.B. Parker, "Self-Assessment Proc. IX: A self assessment procedure dealing with ethics in computing", Communications of the ACM, 25, 3, March 1982, pp. 181-185.

12. D. B. Parker, S. Swope and B.N. Baker, Ethical Conflicts in Information and Computer Science, Technology and Business, QED Information Sciences, Inc., MA, 1990.

13. J. Rest, D. Cooper, R. Coder, J. Masanz, and D. Anderson, "Judging the Important Issues in Moral Dilemmas - An Objective Measure of Development", Developmental Psychology, 10, 4, 1974, pp. 491-501.

14. R.C. Sweetland and D.J. Keyser, ed., "Test Supplement", Test Corporation of America, 1984, p. 57.

15. A. Schlaefli, J.R. Rest and S.J. Thoma, "Does Moral Education Improve Moral Judgement? A Meta-Analysis of Intervention Studies Using the Defining Issues Test", Review of Educational Research, 55, 3 (Fall 1985), pp. 319-352.

16. L. Kohlberg, "The development of children's orientations toward a moral order", Vita Humana 6, 1963, pp. 11-33.

17. R.W. Bickel, M.M. Larrondo-Petrie, D. Bush, "Moral Development and Ethical Computer Use", International Business Schools Computer Users Group IBSCUG Quarterly, to appear 1990.

Courseware Engineering: A Software Engineering Framework for Information Systems Course Development

Daniel Farkas
Pace University

INTRODUCTION

Information Systems has had more explosive growth in its "body of knowledge," than any other discipline in higher education. This has resulted in the development of hundreds of courses and programs to meet our professional responsibilities to students, research, and industry. What hasn't yet emerged, is a sound methodology for developing courses and at the same time satisfying pressure to revise and modify courses in response to technological developments and changing student requirements. Every years individual faculty members must review course contents and departments must evaluate program structures to determine their currency and relevancy in light of new information. This is a costly and time consuming process which could be facilitated by the use of a consistent set of procedures.

It shouldn't be surprising that such a set of procedures exists and that it emerges from within the discipline itself. It is a natural extension of the procedures and principles of the Software Engineering software development process to the course development one. When considered from this point of view, the relationship of course development to Software Engineering is striking. A curriculum (e.g. major) is a "system" and courses are "programs". Not surprisingly, courses consist of "modules", and we are interested in the relationships between curricula, courses, and modules. One can even speak of the relationship between courses and modules in terms of "coupling" and "cohesion". This paper, then, will present a methodology for developing courses which mirrors the System Development Life Cycle,

PRIOR WORK

Prior work in this area is quite extensive, and derives from the private sector. Instructional development targeted towards the formulation of corporate training programs uses design methodologies which incorporate life cycle phases. Recent work exploring instructional design are includes [camp87], [cant88] and [tayl88], [hall88] and [wils88].

A complete discussion of Software Engineering concepts and life cycle models can be found in [fair85], [zelk79] and [lewi82].

COURSEWARE ENGINEERING

Software Engineering disciplines use different models and paradigms for the system development life cycle. While there may be different names and methodologies, Table 1 lists the basic components of system development and the activities associated with course development.

While the level of detail required by software development is not necessary for course development, the overall objectives will be achieved by applying software engineering techniques and adhering to life cycle phases as closely as possible. These objectives are:

Consistency - The order of the modules and their contents should make sense. Topics should not be out of order and relatively self-contained; material should not overlap except in so far as it helps in the presentation of new

material and so on.

Communication - If more than one person is working on the development of the materials, different software engineering techniques account for the communication of information between members of the development team. Furthermore these documents can be used during the revision process.

Maintainability/Modifiability - Since as in all software systems, courses change due to changing requirements, Software Engineering techniques can help in the maintenance and revision of existing courses.

Flexibility - It is desirable that the course be able to adapt to different student populations and semester time constraints.

Utility - The materials for both instructor and students should be manageable within the constraints imposed by the course (prerequisites, time, etc.).

ANALYSIS: CURRICULUM REQUIREMENTS

As in software development projects, it is necessary to determine the requirements for a new course from a variety of sources. Each source will answer questions and provide information which influences the ultimate design of the course:

Students - Who will be taking the course (undergraduate or graduate)? What preparation will they have (prerequisites)? How large will the class be?

Professional - What topics are appropriate for the course? Is there a published recommended curriculum? Is a laboratory necessary?

Institutional - Where does the course fit in the institution's curriculum? Is it required of all students? Majors? Is it part of a sequence of courses and if so what is in the preceding course, and what should the student know before entering the next course?. How much time is available to teach the course (1 or 2 semesters)?

Source Material - Are there books or journals which cover the topic? Have courses been developed? Are there any existing materials (e.g. tests, outlines, handouts).

Answers to the above questions are derived through discussion with colleagues and research in the professional literature. Products of the Analysis phase include: a statement of objectives for the entire course and each of its topics; a topical outline (no order), target audience analysis; bibliography and existing document file.

DESIGN: THE COURSE OUTLINE AND FORMAT

In the Design phase, detailed outlines of the course and course modules are developed. The principles of top-down design and stepwise refinement can be applied since the development consists of designing a modular structure (major topics), and within each module, subtopics -- not too different than a software system. Usage of the terminology and techniques of Structured Design if not formally than informally, clarifies the relationships between topics and the modular structure of the course (e.g. coupling, cohesion, etc.).

During this phase of the course development, it is necessary to decide on the format of the course and how the time is to be divided by both topic and by different instructional technique: lectures, lab, assignments, tests, media (transparencies, video, software).

The decisions made here are very important since the preparation of materials (similar to the implementation of a software design) is difficult and time consuming. The end product of the Design phase would be detailed course outlines and objectives, assignment descriptions, software requirements, textbook selections, media requirements, student evaluation criteria.

IMPLEMENTATION: CREATING COURSE MATERIALS

The creation of course materials is done in the implementation phase. The course complexity will determine how much time is needed for the development of materials. For example, a one-time offered, lecture only, undergraduate course may only require the Design outlines and examinations. Courses which are multimedia obviously require greater effort to produce materials. Table 2, below, contains a list of what might be produced during the implementation phase of the course development. Not all courses will require all the deliverables in the list.

Software is any computer based application which must be created, purchased, installed, or setup for the purpose of student and faculty use. This includes packages for PC's, mainframes, etc. For example, the creation of a Hypercard system would fall into this category. Media includes any type of instructional medium which may require lead time to acquire or create (e.g. film, video tape, etc.).

(TESTING/VALIDATION/VERIFICATION)

Once the course has been developed, it is necessary to test it (like a software system), in order to evaluate whether it is "working." This involves the scheduling of one or two pilot offerings in which both instructor and student feedback (e.g. interviews and evaluations) can be used to modify the contents and organization of the course.

The only way to know whether the course is meeting its objectives is to offer it at least once. Examinations are also appropriate for validation.

MAINTENANCE

In Information Systems, after a course has been in place for a year or two, it is appropriate to consider whether the course requires revision. Revision may result from external (technological evolution) or internal (addition/change of resources) causes. This process is facilitated by having the original Analysis and Design documents available. Updating modules, deleting or adding new ones, is similar to software maintenance. The process iterates through all the phases, with only the new changes taking effect. The course developer can be more responsive to a changing environment.

In situations involving required courses, faculty other than the original developer(s) can easily join the process with minimal effort.

WALKTHROUGHS

Walkthroughs in the software development life cycle, are used to communicate the structure of a system (or program), and thereby uncover flaws in design. A similar process is used in courseware development in which the designer presents an overview of the course. Participants (e.g. chair, faculty, dean) in the walkthrough ask questions about form and content. When done prior to implementation walkthroughs are an effective way insure that the course contents meets the course objectives. It also allows for feedback on the format of the course.

COURSE DEVELOPMENT TOOLS (CODE).

Similar to the current rush toward Computer Aided Software Engineering (CASE) tools, it is natural that a similar development occur for course development. An existing set of analogous computer based tools already exists. It includes word processors, desktop publishing systems, presentation graphics packages, outliners, and associated supporting hardware (e.g. high resolution monitors, laser printers).

With these basic desktop tools and the numerous authoring systems currently available (e.g. Hypercard), the course designer has a vast and powerful development capability at his/her disposal.

APPLYING THE METHODOLOGY

The application of many of the Courseware Engineering techniques presented in this paper has been used recently in the development of an undergraduate course in computer literacy for liberal arts students, and in a sequence of graduate courses in telecommunications for corporate setting. In general the course development process was efficient and flexible.

In the case of the corporate program, there were many supporting resources (secretarial, administrative, hardware and software) which bore the time/cost burden for the faculty/designers. The fact that the program development was a team effort (each course had at least two designers), necessitated some methodology to meet the course offering deadlines successfully.

The undergraduate literacy course was created by a single faculty member and while designed with courseware engineering principles in mind, was less formal in its implementation.

Currently the Information Systems Faculty at the University are looking at the entire IS curriculum. If the methodology is adopted, this will provide a fertile testing ground for these techniques.

BIBLIOGRAPHY

[camp87] Campbell, Clifton P., "Instructional Systems Development: A Methodology for Vocational-Technical Training," Journal of European Industrial Training, (11,5), 1987, pp. 2-42.

[cant88] Cantor, Jeffrey A., "An Automated Curriculum Development Process for Navy Technical Training", Journal of Instructional Development, (11,4), 1988, pp. 3-11.

[dema78] De Marco, T., Structured Analysis and System Specification, Yourdon Press, 1978.

[fair85] Fairley, R.E., Software Engineering Concepts, McGraw- Hill, 1985.

[hall88] Hall, K. A.,"Computer Applications in Education and Training: Curriculum Development Model," Proc. of the Fourth Annual Conference on Microcomputers in Education, R. A. Camuse, ed., 1988,pp. 185-191.

[lewi82] Lewis, T.G., Software Engineering: Analysis and Verification, Reston Publishing Company, 1982.

[tayl88] Taylor, Robin, Doughty, P.L.,"Instructional Development Models: Analysis at the Task and Subtask Levels," Journal of Instructional Development, ((11,4), 1988, pp. 19-28.

[wils88] Wilson, M. A. B.,"Curriculum Development Steps," Proceedings of the Fourth Annual Conference on Microcomputers in Education, R. A. Camuse, ed., 1988, pp. 446-451.

[your78] Yourdon, E., L. L. Constantine, Structured Design, Yourdon Press, 1978.

[zelk79] Zelkowitz, M.V.,A.C. Shaw, J.D. Gannon, Principles of Software Engineering and Design, Prentice-Hall, 1979.

Phase	Course Development Activity
Analysis	Determine the requirements Create curricula objectives
Design	Outline course, modules Decide on presentation formats Choose student materials (e.g. textbook)
Code	Develop materials
Test	Run one or more pilots
Maint.	Revise

Table 1. Course Development Life Cycle

- Lecture Notes
- Transparencies
- Software
- Media
- Instructors Guide
- Student Handouts
- Examinations

Table 2. Implementation Phase Deliverables

A LONGITUDINAL STUDY OF CLASSROOM COMPUTER STRESS

Marilyn K. McClelland
School of Business
North Carolina Central University

E. Holly Buttner
Bryan School of Business and Economics
University of North Carolina at Greensboro

ABSTRACT

Students in a computer training course participated in a longitudinal study of stress associated with learning to use computers. The relationship between stressors, moderators and consequences of learning to use computers is examined. Teacher support and user attitude are found to be related to lower levels of reported stress.

INTRODUCTION

In today's business environment, microcomputers are used by many employees to automate a variety of tasks in the work place. Since schools of business strive to prepare students to be productive in today's business environment, the extensive use of micros in the work place has led to an increase in the use of micros within the business school curriculum. The increase in the use of micros in the work place and the classroom has provoked study of integrating computers into the work place and/or the classroom.

The technical obstacles to implementation have been the topic of recent research. For example, Guimaraes and Ramanujam [6] examined issues including cost, integration of the system, and security. But the problems that arise in implementing new systems are managerial, not technical [4]. Little attention has been directed to the behavioral aspects of implementation efforts. One exception is Liker, Roitman, and Roskies [9] who recognized the impact on the organization and its human resources when technology changes.

Research has shown that the degree of resistance by individuals affected by a change can be a major determinant in the level of successful implementation. Ginzberg [5] showed that user commitment was crucial to successful implementation of a management information system. Individuals may resist the change process because of the stress they experience in the process of implementation. Stress leads to several dysfunctional outcomes for the individual including lower commitment, job dissatisfaction, absenteeism, health problems, and lower productivity [11]. However, there has been little systematic study of the effect of technology change on level of stress. Here, we examine the use of micros in a classroom to gain insight into the interrelationships of technology change and stress levels as indicated by our outcome variables of class satisfaction and health problems.

Brief, Schuler, and Van Sell [2] developed a model of the factors leading to stress. The effect of organizational and job stressors may be moderated by individual attributes including type A versus type B personality [11]. Hence, in this study, we included the personal attributes of personality type and attitude as possible moderators of

stress. In a study of stress reported by information system managers, Weiss [14] found that social support, when present, reduced stress. Therefore, a third possible moderator, social support, was included in the present study. Benner [1] found that previous experience, the fourth moderator in this study, with a particular event led to perceptions that these events were less stressful over time. Thus previous experience with computers may lead students to experience less stress in the class.

The purpose of our study was twofold; first, we examined the interrelationships among stressors, moderators and outcomes in a technology change context where trainees were learning to use computers. Second, we were studying the implementation of a technology change in a specific context, the classroom, to learn how training might be more effectively conducted to reduce stress. A number of scales in a questionnaire were used to measure the consequences of the introduction to and training in using computers for students.

METHODOLOGY

Sample:

Fifty undergraduate students in business administration voluntarily participated in the study. Thirty-two participants were students in a computer course; the remaining 18 were students in a management class which did not use computers. The second class served as a control group. The computer class is a required course for undergraduate business students. The course is an elective for other students in other programs of study in the university. The computer course consists of learning to conceptualize problems, develop flow charts, and write programs to solve business problems. Students generally consider the course to be very demanding.

Procedure:

All participants completed the first of two versions of a survey instrument and the results were compared to determine whether the computer class constituted a representative sample of business students. Computer class students also completed a second version of the questionnaire in the eighth week so that changes in variables over time could be assessed. The survey was administered by the researchers during class time. Students voluntarily completed the questionnaire. Since this was a longitudinal study, responses to the second administration of the survey were matched to a particular individual's responses for the first survey. To maintain anonymity, students were asked to code the survey's with a portion of their student identification number so that responses to the two surveys could be matched. Unmatched responses were discarded. Instructions regarding participation were identical across sections of the course. Anonymity of participants was assured.

Measures:

Stressors included computer stress, class stress and global stress which were measured in three separate scales. Computer stress consisted of items measuring ambiguity, conflict and overload as developed by Rizzo, House and Lirtzman [12]. The items were shortened and modified for this survey. Class stress was a 7-item scale developed by the authors. Finally, Cohen, Kamarck and Mermelstein [3] developed a 17-item scale to measure global stress, a general assessment of stress experienced.

Moderators were measured using six scales. The Framingham Type A 10-item scale was used to determine the respondent's personality type [7]. The student's self-esteem was indicated by a 3-item scale which was a modified

version of House, Wells, Landerman, McMichael, and Kaplan's [8] occupational self-esteem scale. Social support in general and that provided by friends and the instructor were each measured by modified scales of 5 items developed by Schaefer, Coyne, and Lazarus [13]. Students indicated their previous experience with micro computers and whether they owned a computer as measures of previous experience. A student's attitude toward computers was measured using a 5 item scale.

Finally the outcome variables, stress related health problems and class satisfaction, were measured as well. The number of stress related physical symptoms, their frequency and their severity were reported on a 30-item scale [10]. Participants responded to a 5-item scale indicating their class satisfaction [8].

Analysis:

Preliminary data analysis included statistical comparison of the experimental class group to the control class group. The control class did not use micro computers during the course. There was no significant difference between classes for moderators including sex, personality type, major, or social support. Further, there was not a significant difference in global stress or in stress related physical illnesses at the beginning of the semester between the control group and the experimental group. The only statistically significant difference between the classes at the beginning of the semester was that the control group had more upper class students (juniors and seniors) than the experimental group. Pearson correlation coefficients were computed for all variables in each survey and for changes in variables across both surveys. Additional data analysis may be conducted after further review of the preliminary data analysis.

RESULTS

Statistically significant correlations of moderators, stressors, and outcomes are summarized in figure 1. Within the figure we identify responses from the first survey by a 1 and responses from the second survey by a 2. A response marked with a C indicates a change in a response from the first survey to the second survey. For example, in figure 1 we see that high levels of global stress as measured in survey one and stress related physical illnesses are significantly correlated with Type A personalities as assessed in survey one.

In support of stress research in other contexts [7], we found that type A personality was associated with higher levels of stress. Support from the instructor was associated with lower levels of stress and higher class satisfaction, again consistent with Weiss' [14] findings. Students who reported positive attitudes about computers reported lower levels of stress. Contrary to Benner's [1] research, previous experience was not significantly related to lower levels of stress.

DISCUSSION

The purpose of this study was to conduct an exploratory analysis of the relationship between stressors, moderators and consequences of learning to use computers in a classroom setting. Are there any procedures or techniques which can be used to facilitate user adaptation to a technology change? It appears that there are. A teacher's positive influence on student attitudes could help change negative perceptions and resistance to computers.

The results generally supported previous research. Students who were worried about using micros reported higher levels of physical illness. Personality type and attitude were associated with self-reported stress in a technology

change context. However, previous experience was not related to lower levels of stress. Perhaps future research could explore this relationship.

FUTURE RESEARCH

Additional study of the data may provide insights about sources of stress and ways to moderate stress. An increase in understanding computer stress may lead to the development of instructional techniques that reduce computer stress. An extension of this research would be to conduct a similar study in a firm implementing a new technology such as office automation to examine the stressors, moderators, and outcomes in an industrial setting.

authors' note: This paper was condensed to adhere to the proceedings page restrictions. A version of the paper with additional research details is available from the authors.

REFERENCES

1. Benner, P. Stress and Satisfaction on the Job. New York: Praeger, 1984.

2. Brief, A., R. Schuler, and M. Van Sell. Managing Job Stress. Boston: Little, Brown, and Co., 1981.

3. Cohen, S., T. Kamarck, and R. Mermelstein. "A Global Measure of Perceived Stress," Journal of Health and Social Behavior, 1983, 24, 385-396.

4. Curtain, F. T. "Planning and Justifying Factory Automation Systems", Production Engineering, August 1984, 46-51.

5. Ginzberg, M. "Key Recurrent Issues in the MIS Implementation Process," MIS Quarterly, 1981, 5, 47-56.

6. Guimaraes, T. and V. Ramanujam. "Personal Computing Trends and Problems: An Empirical Study," MIS Quarterly, 1986, 10, 179-187.

7. Haynes, G., M. Feinleib, and W. Kannel. "The Relationship of Psychsocial Factors to Coronary Heart Disease in the Framingham Study. III: Eight Year Incidence of Coronary Heart Disease," American Journal of Epidemiology, 1980, 37-58.

8. House, J., J. Wells, L. Landerman, A. McMichael, and B. Kaplan. "Occupational Stress and Health among Factory Workers,"Journal of Health and Social Behavior, 1979, 20, 139-160.

9. Liker, J. K., D. B. Roitman, and E. Roskies "Changing Everything All at Once: Work Life and Technological Change," Sloan Management Review, Summer 1987, 29-47.

10. Matteson, M. and J. Ivancevich. "Note on Tension Discharge Rate as an Employee Health Status Predictor," Academy of Management Journal, 1983, 26, 540-545.

11. Quick, J. and J. Quick. Organizational Stress and Preventive Management. New York: McGraw Hill, 1984.

12. Rizzo, J., R. House, and S. Lirtzman. "Role Conflict and Ambiguity in Complex Organizations," Administrative Science Quarterly, 1970, 15, 150-163.

13. Schaefer, C., J. Coyne, and R. Lazarus. "The Health-Related Functions of Social Support," Journal of Behavioral Medicine, 1981, 4, 381-406.

14. Weiss, M. "Effects of Work Stress and Social Support on Information Systems Managers," MIS Quarterly, 1983, 7, 28-43.

Table 1. CORRELATED STRESSORS, MODERATORS, AND OUTCOMES

1:Type A personality	1:High level of global stress*** 2:Many stress related physical illnesses*
1:High GPA	2:Not worried about using the micro* 2:Few sources of stress within the class* 2:Perceive lack of friend support* C:Less worried about using the micro*
1:High level of global stress	1:Type A personality*** 1:Upper level students (juniors or seniors)* 2:Dissatisfied with the class* 2:Many stress related physical illnesses** 2:Perceive a lack of instructor support* 2:Many sources of stress within the class*
C:Increased level of global stress	C:Perceive less social support*
1:Worried about using the micro	1:Negative perception of micro (not useful)** 1:Female** 2:Negative perception of micro*** 2:Dissatisfied with the class** 2:Many stress related physical illnesses* 2:Many sources of stress within the class** 2:Worried about using the micro***
2:Worried about using the micro	1:Low GPA* 1:High levels of global stress* 1:Not an IS or OM major* 1:Worried about using the micro*** 2:Many sources of stress within the class*** 2:Dissatisfied with the class*** 2:Negative perception of micro** 2:Feel supported by friends* 2:Many stress related physical illnesses* C:Perceive less significant other support* C:Perceive less instructor support* C:Increase in worries about using the micro***
C:Increase in worries about using the micro	1:Low GPA* 1:Male* 2:Worried about using the micro*** 2:Many sources of stress within the class** 2:Dissatisfied with the class*
1:Positive perception of micro (useful)	1:IS or OM major*** 1:Not worried about using the micro** 2:Positive perception of micro*** 2:Satisfied with the class** C:Perceive more significant other support*

Table 1. CORRELATED STRESSORS, MODERATORS, AND OUTCOMES-continued

C:Enhanced perception of micro	2:Positive perception of micro***
	2:Satisfied with the class***
	2:Few sources of stress within the class***
	2:Not worried about using the micro**
	2:Perceive instructor support*
	C:Perceive more instructor support*
2:Perceive friend support	1:Low GPA*
	2:Worried about using the micro*
	C:Perceive less instructor support*
	C:less frequent physical stress symptoms*
2:Perceive instructor support	1:Low level of global stress*
	2:Satisfied with the class***
	2:Fewer stress related physical illnesses*
	2:few sources of stress within the class**
	2:Positive perception of micro*
	C:Perceive more instructor support***
	C:Enhanced perception of micro*
	C:Perceive less relative support*
C:Perceive more instructor support	2:Perceive instructor support***
	2:Satisfied with the class***
	2:Few sources of stress within the class**
	2:Perceive lack of friend support*
	2:Not worried about using the micro*
	C:Enhanced perception of micro*
2:Many sources of stress within the class	1:Not an IS or OM major**
	1:Low GPA*
	1:High level of global stress*
	1:Worried about using the micro**
	2:Dissatisfied with the class***
	2:Worried about using the micro***
	2:Negative perception of micro***
	2:Perceive low instructor support**
	C:Heightened negative perception of micro***
	C:Perceive less instructor support**
	C:Perceive less significant other support**
	C:Increase in worries about using the micro**

1: indicates measure at the beginning of the semester
2: indicates measure at mid-semester
C: change in the measure between the beginning of the semester and mid-semester

*** correlation significant at the .001 level
** correlation significant at the .01 level
* correlation significant at the .05 level

TRAINING AND EDUCATION IN INFORMATION SYSTEMS: AN ISSUE RESOLVED

Susan M. Moncada
Indiana State University

Karen J. Ketler
Eastern Illinois University

A recurrent issue in higher education is the proper role of training and education in preparing students to enter the work-force as professionals. This paper explores the meaning of "training" and "education." It suggests that teaching methods should be employed that best suit the goals of individual courses. How one university has incorporated both training and education into an introductory computer class is explored.

With the proliferation of the microcomputer and microcomputer application software, end-user computing has exploded during the 1980's. Training and education provide current and prospective employees with the necessary skills required to effectively use the computer as a tool. For example, a survey of the Illinois Banking Industry determined that employees learned how to use various computer applications through in-house computer training (87%), vendor training (69%), and schools (37%) [11]. Currently, "it is training, not formal education that provides the skills for two out of three jobs" in this country [1].

A recurrent issue in higher education is the proper role of training and education in preparing students to enter the workforce as professionals. This issue affects not only the preparation of information systems professionals but others who will eventually be end-users. Typically, education is associated with concepts and theory, while training is associated with skill development. Purists are intent on distinguishing between education and training.

This paper explores the meaning of "training" and "education." The education vs. training dilemma in academia may very well be over-rated. The real question should be how best to prepare students for entry into the professional work-force. Academia has been repeatedly criticized for doing this inadequately, particularly in the computer field [8,9,12]. Both the DPMA and ACM have developed model curricula to guide the content of majors in the computer fields. Teaching methods should be chosen that best suit the goals and objectives of individual courses.

EDUCATION VS. TRAINING

Schools, colleges, and universities essentially provide learning to non-employees. The goal of traditional academia is to educate. It teaches concepts and the evolution of those concepts. It teaches critical methods of inquiry and knowledge in various subjects. It teaches historical background as well as current issues. Finally, higher education supplies the foundation for many professions [3].

Other providers of learning include consultants, corporate schools, professional organizations, and human resource departments within companies. These organizations essentially provide training to employees. The learning opportunities provided by these organizations are more specific and

narrow in terms of applicability to the company's needs and productivity. Training is characterized in the corporate world as being short-term, labor intensive, and having highly motivated employee-students.

According the Webster's New World Dictionary, education is "the process of training and developing knowledge, mind, character, etc., especially by formal schooling at an institution of learning." Synonyms for education include: learning, instruction, knowledge, scholarship, schooling, literacy. Training, on the other hand, is "to subject one to certain actions, exercises, etc. in order to bring to a desired condition; to instruct so as to make proficient or qualified." Webster's definitions suggest that training is a subset of education. Thus, training appears to be more specific than education.

HUMAN RESOURCE DEVELOPMENT: EDUCATION AND TRAINING

In the business world, human resource development has become an umbrella for the various forms of employee learning opportunities provided. Human resource development, adopted by the American Society for Training and Development is defined as "organized learning experiences in a definite time period to increase the possibility of improving job performance and growth" [7]. The following analysis of this definition provides insight into the training/education issue in information systems.

Organized learning experiences, refer to "deliberate learning situations that have planned objectives to be accomplished and conditions for evaluation" [7]. The behavioral objective consists of three parts: a condition statement, an expected performance, and a standard by which to judge performance. These objectives guide learning. They act as goals for both the instructor and student. The instructor knows what needs to be accomplished and has developed precise expectancies. The students know what is in store for them and how they will be required to perform.

A **definite time period** refers to "an agreed upon and identifiable point at which a particular phase of learning has been completed" [7]. On this element academia and industry differ. At the college or university level the time period involved is generally long. For example, it might be a 16 week semester. Students attend a class 2-4 hours per week. In business and industry the time period is extremely compressed. Training is intensive, perhaps for one week, eight hours per day.

Increasing the possibility of improving job performance acknowledges the fact that the provider of learning situations can't guarantee that these experiences will change performance. Other factors such as, motivation, learning style, intelligence, etc., affect what individuals grasp. The "teacher" can, however, attempt to make learning experiences deliberate positive opportunities for the participants.

Improving job performance and providing for employee **growth** are the goals of a human resource department. It encompasses three types of learning activities: training, education, and development. In the business world **training** involves those activities which are designed to improve the performance of employees in their current job. The **education** of employees refers to those activities which are designed to improve the overall competence of employees in a specified direction beyond their current job. The **development** of employees is concerned with preparing employees so that they can move with the organization as it develops, changes, and grows [7].

The American Society for Training and Development defines training and development as "identifying, assessing,

and through planned learning, helping develop the key competencies which enable individuals to perform current and future jobs" [6]. Both the ACM and DPMA developed model curricula to do just this. The common thread between training and education is learning. Clearly at times, educators need to train and trainers need to educate.

LEARNING REVISITED

Learning is the process of acquiring skills, knowledge and/or attitudes. There are three learning domains, the cognitive, affective, and psychomotor. The cognitive domain involves gaining knowledge or understanding. It refers to acquiring ideas, principles, concepts, or facts. The affective domain involves the acquisition of attitudinal values. It is concerned with enhancing interests, appreciations, and ideals. The psychomotor domain is behavioral. It involves the acquisition of abilities or skills, both mental and physical. Behaviors include habits or ways of doing something.

Learning is accomplished through the five senses: seeing, hearing, feeling, smelling, and tasting. In planning any learning activity it is important to remember that the five senses are the channels through which individuals are stimulated. The teaching method chosen should facilitate the result sought. Figure 1 [5] suggests sources to use in order to achieve desired results:

FIGURE 1

* PROMOTE UNDERSTANDING WITH INFORMATION USING
  (1) Articles
  (2) Audiotapes
  (3) Diagrams
  (4) Lectures
  (5) Programmed Instruction

* PROMOTE SKILL ACHIEVEMENT THROUGH EXPERIMENTATION USING
  (1) Case Studies
  (2) Demonstrations
  (3) Exercises
  (4) Role Playing
  (5) Video Tapes
  (6) Worksheets

* PROMOTE DIVERSIFICATION THROUGH INQUIRY AND OBSERVATION USING
  (1) Exercises
  (2) Films
  (3) Instruments
  (4) Role Playing
  (5) Self-Analysis
  (6) Structured Games

* PROMOTE CREATIVITY BY EXPERIENCING INNOVATION USING
  (1) Brainstorming
  (2) Exercises
  (3) Mental Acuity
  (4) Self-Analysis
  (5) Unstructured Games

A TRAINING/EDUCATION APPROACH

Eastern Illinois University has wrestled with the training vs. education dilemma in its Introductory Computer class, "Computer Systems and Microcomputer Applications" (BED 2510). The course has two basic goals: (1) to provide an opportunity for students to learn the basic skills essential for becoming computer literate, and (2) to enable them to use the computer as a valuable tool throughout their college life and beyond. Essentially eight weeks of theory is taught. Topics covered include: hardware components and systems, data communications, computer ethics and the impact of computers on society, and prewritten software packages for business. The remaining time is devoted to teaching DOS and three application packages: WordPerfect, Lotus, and RBase.

BED 2510 is a multi-section class with enrollment of approximately 1400 students per year. Total enrollment at the university is approximately 10,000

students. The typical class size is 35-40 students. It is open to all majors and can be taken at the Freshman level. The course must be completed by all pre-business majors prior to being admitted to the College of Business. There are no prerequisites, although keyboarding is recommended. As a result no MIS is covered in the class. Instead, all business majors must complete a separate, senior-level class dedicated to Management Information Systems.

To ensure that uniformity exits among sections, a common syllabus is used. Two BED 2510 coordinators are responsible for monitoring the course and ensuring that guidelines are followed. The class is taught by tenure-track faculty and full or part-time instructors.

Originally the course was taught in the traditional classroom with all hands-on computer use done outside of class time. No computer lab/classrooms were available. Lectures dominated as the primary teaching methodology used. Lectures were supplemented with the use of overhead transparencies. Occasionally a computer would be brought into the classroom for demonstration purposes. This environment was far from ideal. No classroom training was provided. Evaluation consisted of written exams and the completion of homework assignments.

As new equipment became available and computer classrooms were created, teaching methods evolved. Now, the BED 2510 classroom is equipped with a video display unit. This device enables the contents of the computer monitor to be projected on a screen. As spreadsheet concepts are discussed, students are given a preview of what they will see when they use the software by themselves. Feedback has been extremely positive. When given the choice of teaching methods, students prefer this type of demonstration to the traditional chalkboard lecture or transparency lecture.

In addition two computer laboratories were installed. One room is equipped with a video display unit. The availability of this equipment has allowed faculty to incorporate training as a viable teaching methodology. The labs act as instructional support environments for more adequate teaching of microcomputer skills. To ensure that the emphasis on theory and concepts is not sacrificed, these labs may be scheduled for classroom use by BED 2510 faculty for only 40% of the class meetings.

In BED 2510 students are eased into using a microcomputer. Regular classroom demonstration followed immediately by supervised practice in the lab is most effective. The instructor assumes the role of a coach. Computer anxiety is minimized. As a result, students are set up for success. Using the computer becomes a positive experience.

Faculty also use the lab facilities to test students' microcomputer skills. Students appear to be more motivated when they know they must demonstrate their abilities. They are less inclined to copy one another's homework assignments.

Like the majority of other introductory level computer classes [10], BED 2510 focusses on the attainment of knowledge, comprehension, and the application of skills. When used in the proper context, making training a part of the education process can be an effective teaching technique.

BIBLIOGRAPHY

1. Carnevale, A.P. "The Learning Enterprise." Training and Development Journal, January, 1986.

2. Egelston, Diane C. "The Big Business of Employee Training." Business Week, March 1986, pp. 80-82.

3. Eurich, N.P. Corporate Classrooms. Princeton: Carnegie Foundation, 1984, pp. 1-22.

4. "Human Resource Development: A Useful Bit of Jargon?" Training, January 1985, pp. 75-76.

5. Leach, J., Training Programs in Industry, VOTEC 353, University of Illinois, Spring 1986.

6. "Models for Excellence." American Society for Training and Development, Washington D.C., 1983.

7. Nadler, L. The Handbook of Human Resource Development, New York: John Wiley & Sons, Inc., 1984.

8. Ridgeway, M. "Curriculum Shortfall." Datamation, December 15, 1988, p. 77.

9. Rifkin, Glenn. "Top Students Shunning MIS, "Computer World. June 15, 1987, pp. 120-121.

10. Spence, J.W., and Windsor, J.C. "The Teaching Process: Training vs. Education," Interface, Summer, 1988, 10:2-5.

11. Wentling, R. M., and Wentling, T.L. "What Illinois Institutions are Using." Bank Administration. January, 1988, pp. 42-45.

12. Yaffe, J. "Mis Education: A 20th Century Disaster." Journal of Systems Management. April 1989, pp. 10-13.

TESTING FOR MICROCOMPUTER COMPETENCY

David R. Callaghan
Bentley College

ABSTRACT

Many colleges and universities have implemented microcomputer courses which have as a major component the introduction of application software. Problems arise when attempting to test student competency in these areas as hands-on testing is vital. This paper presents a methodology developed and tested by the author which can be done in small groups over extended periods.

INTRODUCTION

At least one course in computer technology is available at virtually all colleges and universities. Many of them are required for at least some portion of the student populations and most involve large components on microcomputer applications. Word processing, spreadsheets, and frequently database are considered by many to be core packages in today's college environment.

Testing students, while an accepted standard for traditional memorization-based lectures, is difficult for applications which are interactive with machines. Limitations on machine availability, space and grading complexity make hands-on testing a more complex environment.

The author has developed and tested a hands-on testing strategy at which has been extremely successful. This methodology should be easily transferred to other institutions, regardless of their equipment configurations.

HANDS-ON TESTING AT BENTLEY

Bentley College is an AACSB accredited private business school of roughly 3500 undergraduates. Since the 1970's, it has had two required computer courses in its curriculum. In 1985, after a pilot test, the college decided to require all students to have a portable personal computer, beginning with that year's freshman class. Students must take the Principles of Computing course the first semester they come on campus, concurrent with the receipt of a computer.

Roughly 1100 freshmen and transfers sign up for one of 28 lecture sections each fall. It is each faculty member's prerogative as to how many exams and what homework assignments to give in class. The final exam is comprehensive and departmentally based, given in one time slot during final exam week. Relative weights for each component of the final grade are suggested, but not mandated.

In 1986, a non-credit, 75 minute laboratory component was added. Students register for one of 56 laboratory sections (20 person maximum), in addition to the two 75 minute lectures. Lab sections contains students from several different lecture sections.

Students bring their machine to a classroom where directed exercises take place under the tutelage of graduate assistants. The same exercise is done in all sections. Graduate assistants do not teach. Their role is to help students overcome difficulties as they

complete the exercise.

In 1987, the hands-on exam for the laboratory was instituted as the last lab of the semester (the week before finals). After three years of testing, we consider it very successful.

The Bentley Laboratory Exam

An objective of the course and the exam is to ensure a minimum knowledge of operating systems (MS-DOS), an introductory level of word processing (WordPerfect), a fair understanding of spreadsheets (Lotus 1-2-3). We also expect students to be able to transfer data between packages. The test is designed to capture all these components.

There is no database component in this first course. Database and modeling are two major topics of the MIS course taken in the junior year.

With 56 lab sections offered in 28 periods during the week, we needed to control for inter-sectional learning as the week progressed. To this aim the author developed a single form upon which the 28 different exams are based (Appendix I).

There are ten data sets for step 2 and the other steps are semi-randomly chosen. A fully random assignment is not used as it could penalize students with multiple techniques they often find difficult. For example, we avoid exams which contain more than two of the following: formatting and installing the operating system, creation of a subdirectory, @IF functions, labeled pie charts, right justified text and bolding a column.

A sheet outlining the expectations (Appendix II) is given to all students the week before the exam. This avoids variations in what students are told about common exams when multiple faculty provide verbal instructions. It also eases student anxiety regarding the exam's coverage. All exams are made in strict conformance with this outline.

The assignment of specific forms to sections is consciously done. An objective is that students find out quickly that the exams are different although similar. To this end we use the most dissimilar exams at the beginning of the week. By Monday afternoon, the word is out that the exams are different.

When students complete the exam, they hand in disks rather than printed output. We do make the effort to collect exam papers but recognize that some will slip out. Given our testing methodology, we do not consider this a problem. No additional data beyond what we have presented in the student outline can be gained by having a copy of one form of the exam.

Good students are able to complete the exam in 20 minutes. The bulk leave around 40 minutes, with 90% done within an hour. Those left at the end of the period are having so many problems with the exam that they are unlikely to finish if given all day.

Grading Methodology

Grading is done by the graduate assistants who teach the labs. To insure consistency, a detailed point allocation scheme is provided. Each section/exam form is graded using a three stage procedure on two computers. One machine is used to do the grading, the other to enter grades directly into a spreadsheet.

The first stage is to check the MS-DOS step. This checking procedure is automated with a batch file kept on disk with the appropriate procedures in the A: drive. The batch file is given a one letter name to minimize how much the graduate assistant would have to type.

MS-DOS batch file:

CHKDSK B:	check for existence/lack of operating system
DIR B:	check for proper file names
DIR B:\subdir	for exams using subdirectories

Since all disks are formatted in the exam's first step, the only files that should be on the disk are those which are from the exam. It is important that the filenames from steps seven and eleven be proper so later macros will work. When they are improper, the grad assistant copies them (to protect the student's original work) to the proper name on the same disk.

A similar procedure is used for grading the Lotus section. In this case, a macro is created, hidden far away from the home position. Rather than /File Retrieving the student's spreadsheet, we use /File Combine. This prevents the macro from being destroyed each time a student spreadsheet is loaded.

Lotus load macro:

/RE A1.Z256	so you don't erase the macro
{goto}A1~	
/FCCEFORMx~	so the macro doesn't get replaced
/WGFT	so formulas appear on screen

After checking the formulas, the graph is verified by hitting {F10}.

For the WordPerfect section, a keyboard macro is used. Like the single letter batch file for the MS-DOS section, this minimizes keystrokes.

WordPerfect load macro:

{F7}(exit) N(o save) N(o exit)
to clear current workspace
{shift}{F7}(retrieve)FORMx{return}
to load the document
{ctrl}{right arrow}
to get to first char in text
{alt}{F3}(reveal codes)
to see bolding, etc.

Graduate assistants were initially concerned on the amount of grading that they would have to do given their own preparation for finals. Each year their concerns quickly dissipates when they find they can grade an entire section (20 students) in roughly 40 minutes.

Evaluating the Success

We were surprised by the lab exams success the first year. As we had anticipated, results were bimodal, with most students scoring in the high B to A range. This is to be expected given the strict adherence to the outline provided before the exam.

We were even more satisfied that when comparing average scores by lab section, we could find no differences. This indicated both fairness of the exam and that differences in graduate assistant's lab styles did not influence a student's ability to learn the software.

Looking at the data by lecture instructor has resulted in some significant differences. We conclude that this is due to the emphasis that various instructors place on the training component of the course.

PROBLEMS

Problems are pleasantly rare. The most concerning is disk failures and file corruptions which make grading difficult and sometimes impossible. With 1100 disks coming in which are used extensively all semester, we have to

expect some failures. We use Norton Utilities and other tools to recover what we can.

We've also had a few students who hand in the wrong disk. These students and others with unrecoverable disk errors, are asked to retake the exam during a make-up period. There is very little reluctance from students in this regard. Since they knew it the first time, they don't have any additional preparation. They may also fear the alternative (failure), although we never present this as an option.

CONCLUSION

Hands-on finals are not only possible, but manageable. Like any exam, it cannot test the entire knowledge base that students should possess. By selecting the most important topics, we can develop a reasonable mechanism to ensure a minimum level of competency.

The exam methodology described has been well accepted by students who appreciate the consistency that a departmental exam provides. The results have also proven useful in response to upper division faculty who expressed concern that the course never taught the students how to use the computer. Since the exam was instituted, this has not been an issue.

APPENDIX I:
EXAM FORM OUTLINE

FORM x

1. Format the CS101 data disk (with or without) the operating system on it. (Make a subdirectory or no subdirectory.) Copy a specified file from another disk into the (new subdirectory or root directory).

2. Please enter the data below into the upper left corner (starting in A1) of a Lotus spreadsheet:

   This area contains a small (roughly 4 by 5) matrix of data and specifications for some sort of numerical manipulation of that data.

3. Create a graph (line, pie, bar, stacked bar) of some portion of the data points in step 2.

4. Provide appropriate labels along the X-axis. (Note: it is up to the student to determine what cells should be used for this labeling process.)

5. Put your name as a title for the graph.

6. Print the spreadsheet to a file named **FORMxA** on the data disk.

7. Save the spreadsheet as **FORMxB** on the data disk.

8. Exit Lotus and enter WordPerfect. Load the spreadsheet you printed in step 6 into WordPerfect.

9. Put your name (above, below) the data from the spreadsheet and (left justify, center, right justify) it between the margins.

10. (Bold, Underline) some portion of the data in the spreadsheet (row or column).

11. Save the document as **FORMxC.WP** on the data disk.

12. Make sure your name, lab day and time, and your lecture instructor's name is on the outside of the disk.

APPENDIX II:
STUDENT OUTLINE FOR LABORATORY FINAL

1. You must be able to use MS-DOS as follows:
   - FORMAT a disk with or without the operating system (i.e. make it a bootable disk or simply a data disk)
   - create a subdirectory with MD (MKDIR)
   - use the COPY command to copy a specific file from one disk to another or to a subdirectory

2. You must be able to accomplish the following in Lotus:
   - enter numerical and label data
   - take a mathematical formula and put it in a proper form, including @SUM, @AVG and @IF functions
     note: LOOKUP is <u>not</u> required
   - "print" the spreadsheet (or a portion of it) to a file
   - create a graph (line, bar, stacked bar or pie) based on the data above
   - put labels on the x axis hash marks or pie slices
   - put titles on the graph (above the graph, x and y axis)
   - save the spreadsheet and associated graph

3. You must be able to accomplish the following in Word Perfect
   - read an ASCII file (such as the Lotus .PRN file)
   - be able to enhance any portion of the text
     - underline
     - bold
     - center
     - flush right
   - save the file to disk

4. **YOU MUST BRING THE FOLLOWING TO THE EXAM:**
   - Your computer
   - the MS-DOS diskette from the manual which contains <u>all</u> the MS-DOS external commands
   - spreadsheet program (JOE or Lotus) diskette
   - Word Perfect diskette AND TEMPLATE
   - your CS101 class diskette with all of the files on it (do <u>not</u> FORMAT it ahead of time - your exam may require inspection of existing files on the diskette).

     NOTE: the disk will be used to store the files you create and will be handed in to the lab instructor. If you have any personal files on the disk, they should be removed or copied to another disk as backup. On the outside of the disk put your name, lab day and time, and your lecture instructor's name.

5. You may bring MANUALS, including your CS101 Laboratory Manual, for reference. You can write all the notes you want inside the manual but no other papers will be allowed. WARNING: If you have to spend a lot of time looking things up in the manual, you will most likely run out of time during the lab. You will be better off if you learn how to do each of these things without looking.

COMPUTER TRAINING FACILITIES FOR SENIORS

Pauline L. Kartrude
Fred G. Harold
Florida Atlantic University

ABSTRACT

A discussion of the potential for productive microcomputer use by "seniors" (older adults) is presented, followed by a review of selected literature relating to the learning styles of seniors. Results of training courses conducted for such audiences at the authors' university are described, and prescriptive recommendations for the physical environment, the method of presentation, and specific instructional styles and materials are given.

I. INTRODUCTION

The "graying of America" and the microcomputer revolution are compatible, though seemingly distinct, phenomena. We have at present in our society an increasing number of older adults whose minds remain active--though perhaps somewhat slowed--while their bodies diminish in strength and coordination. The increasingly sedentary life of these individuals restricts the activities in which they can participate. Reading is an acceptable pastime, as is television (although the passive nature of these activities precludes the stimulation possible only through interaction). The availability of increasingly "friendly" personal computers and the growing diversity of software packages offer an opportunity for seniors to spend significant blocks of time in stimulating, interactive computer dialogues.

This is not a new concept. The success of SeniorNet, an interactive computer network and associated support system launched in 1986 with older adults as its constituency, indicates the validity of this idea. [KEA] Our intent is to refine and focus the already recognized linkage of computers and seniors by proposing the course structure and the type of facility, both tailored to their unique needs, which will make the learning which precedes independent computer use by adults in the 65-plus age range most effective and comfortable.

The following recommendations for the structure and conduct of initial computer training for older adults are based on both a survey of selected literature and experience in presenting such training for members of the Lifelong Learning Society at Florida Atlantic University in Boca Raton, Florida. The primary objective of this paper is specification of a total computing environment providing hands-on training to older adults who need either (a) an introduction to computers in general or (b) instruction in using a specific package.

Based on previous research and training sessions conducted at our facility, plus our concern for meeting the needs of our audience and delivering the training effectively, we implemented the following policies: A maximum of eight students are to be trained at one time, assuming a single instructor. Daytime class sessions 2 1/2 hours long are to be scheduled to balance commuting time requirements and attention span realities. Initial needs surveys and pre-tests of students are to be complemented with post-tests and follow-up surveys.

II. ADULT LEARNING THEORY

Research suggests that seniors "can pro-

fit from the use of strategies and techniques designed to address their unique abilities and learning needs [such as]: ...slower pacing, use of advance organizers, ... positive feedback, reducing diversity and interaction of tasks, and building on previous experience and knowledge."[PET, 17]

The physical changes which occur as people age affect the learning process, causing "...slowed reactions, substantial decline in visual and auditory perception, and reduced levels of energy and physical stamina" as well as "other noncognitive factors (heightened anxiety in a learning experience, increased susceptibility to interference, motivation to learn)."[PET, 19]

> By designing the learning situation so that the deleterious effects of noncognitive variables are minimized, the teacher can help the older learner overcome these deficits and approach more closely his or her true learning potential. Specific examples of these actions can be seen in the use of accessible facilities, appropriate lighting, selection of content, creation of a supportive environment, and relating new content to older experience. [PET, 71]

> ... older people should be helped to avoid errors to the extent possible. Since they tend to remember errors and repeat them, it is most advisable to design the situation so the successful completion of the task is likely. [PET, 89]

> Ideally, the classes should be offered in a credit/no credit format, with methods of evaluation designed to minimize anxiety and competition for those students who would like credit. Actually, credit and grading are irrelevant to most elders, as it is the intrinsic motivation for learning--the acquisition of knowledge or skills for their own sake-that far outweighs the motivation to learn for any extrinsic reward...And it is this intrinsic motivation of the learner that makes senior adult education distinctive and gives it its great excitement and challenge.[YEO, 6-7]

The following is a selected summary of pointers to encourage learning in older adults, excerpted primarily from Sakata, who drew his ideas from a number of other researchers:

> Avoid over- and under-arousal...eliminate anxiety or stress...by familiarizing the learners with the environment and the task...Ample time should be allowed to master new tasks...A warm and accepting learning environment is also extremely important in reducing over-arousal... provide as much support, praise, encouragement, and assistance as necessary... altering seat arrangement to achieve maximum student contact (e.g. horseshoe or semicircle) may also be helpful to increase stimulation.[SAK, 12-13]

[Note: To see the computer's projected screen image from an angled view would require the seniors to move their heads and bodies more to switch among viewing projection screen, keyboard, and CRT; however, this is a possible area of study (delivery of the same material in different classroom formats)].

> "... the appropriate organizing principle for sequences of adult learning is problem areas, not subjects" (writing better business letters, NOT composition 1).[KNW, 54]

Modified adult learning theories can be applied to training of seniors. As individuals age, the body's physical changes require adjustments in the way material is presented. The materials should be consistent: (What is presented in video or lecture form is organized the same as the prepared notes and handouts; material different from what has been presented will be an 'interference', not

a new learning experience). The information should be presented in the small 'chunks' appropriate to adult learners. Time for assimilation and practice must be adequately provided (seniors need a reduced pace). Positive feedback and reinforcement should be consciously included in the curriculum. Presentation of new material should interact with (and build on) the previous information. All tasks should be designed for success, since older students tend to remember and repeat the errors they made. Grading should not be a part of the process. The senior learner should leave the course with a rewarding feeling of accomplishment. Even though senior students have intrinsic motivation for attending a course, some form of 'reward' should be considered. It could range from a printout that they can take home to show what they have done to an elaborate certificate of completion awarded at the end of the course.

III. FACULTY

Most formal education courses focus on how to teach children and younger adults (who have entirely different reasons for being in class than do seniors). There is a definite need to choose the instructors for this program very carefully. "...recruiting and training of knowledgeable, sensitive, enthusiastic faculty...is the most important task of the coordinator"; the teacher must demonstrate a real desire to teach older adults and an understanding of gerontology. [YEO, 6]

If instructors are not chosen carefully "...the high probability of faculty being assigned to teach people whose needs they do not truly understand - same ageist attitudes and stereotypes as the larger society" [YEO, 5] can result in a lack of confidence by the seniors in their own learning ability. "... there are some distinctive concerns and characteristics of old age that need to be integrated into the basic structure and philosophy of elder education"--specifically: (1) participation and (2) self-actualization. [YEO, 6]

In a hands-on computer training program, the need for participation is easily met by the projects that the seniors do on the computer. It is also necessary to permit the seniors to interject questions, thoughts and personal experiences into the lecture/discussion while the instructor guides the flow of conversation, covers the material, and shows vocal students that they have made important contributions and that their questions were not 'stupid'.

The structure of computer projects/assignments is central to the success of the program. Projects must be designed so that the seniors are in control and can work at their own pace, pushing themselves as little or as far as they wish to go. The projects should be designed for 'non-failure' (designed to minimize possible mistakes as they go, since they tend to remember what they did but not that what they did was wrong; this is different from some learning theories for younger individuals that emphasize 'learning through your mistakes'). The seniors need to feel in control of the projects. In addition, the projects should take into consideration the interest area(s) of the senior (historically, the largest number of requests have been for stock tracking programs, so that the splitting and fluctuations of the stock could be followed).

IV. PACING

When a task has a time limit for completion, the older learner does not perform as well as the typical younger student. "... when self-pacing by the older learner is allowed, the learning performance appears to be optimized." Instruction should be self-paced whenever possible and at a reduced pace otherwise. A lecture is considered a timed instruction process; material should be "presented, reviewed, and examined". "The importance of con-

trolling the pacing of instruction cannot be overemphasized." [PET, 80]

Time limits imposed on the senior can inhibit learning. The older learner may become intimidated, frustrated, and 'turned off' when made to feel that a task must be completed in a given amount of time. Therefore, available self-paced tutorials should be incorporated into the instruction process. With the small number of seniors to be taught at any one time, self-paced tutorials with individualized assistance are feasible. In addition, where handouts contain more than any individual can finish in a given session, there should be 'completion units' within the handout so that the senior can have a feeling of accomplishment upon completion of a sub-project. With this structure, those who do not finish the handout rapidly are not rushed by those who do.

Sessions should be spread over several days or weeks rather than concentrated in a single session lasting several hours. This format takes into account the need for time between tasks. Also, the older learner's reduced stamina and strength do not permit the speed or intensity typical of other learning environments. Two hours of actual class time should be the maximum (plus time allowance for a break for walking, redirecting their attention, and having a snack). Sufficient time for restroom breaks should also be considered, "since older people find it necessary to relieve themselves frequently." [PET, 105]

In accordance with Peterson's statements, classes for seniors should be as follows:

(1) one hour of classroom activity covering:
- a review of the last class
- coverage of new material, relating it to what the learners have previously encountered and how they are going to make use of it
- summary of what has been presented in this session

(2) a break (announce 15 minutes-- they may take a little longer depending on the facilities available)

(3) one more hour of classroom activity:
- a re-review of the first hour, clearing up any questions
- self-paced instruction reinforcing what was taught in the first hour
- prior to ending the session, a summary of the day's information

Classes scheduled every day for a week will provide better retention of material than once a week for five weeks. The senior learner will forget what has been introduced in one week's time unless there is an opportunity to practice between classes. However, transportation problems and prior commitments of the seniors must be considered before implementing a full week's schedule.

> Transportation is a major problem for many older adults, so decentralized programs in senior centers, nutrition sites, apartment complexes, or churches are frequently more convenient than a college or school campus. In any building, the classrooms need to be close to parking and bus-stops, barrier-free, well-lighted, with comfortable seating and minimum noise distraction... Older adults almost unanimously prefer daytime classes, and once-a-week scheduling minimizes transportation obstacles. [YEO, 6]

V. ENVIRONMENT

Temperature. Rooms that are too hot or too cold have an adverse effect on older learners - (They have a preference for temperatures "in the mid to upper 70s"). [PET, 105]. For non-seniors, temperatures are generally kept more toward the lower 70s. In fact, some trainers even prefer the upper 60s to

keep people's minds from wandering [MUN, 110], but this does not seem to be the case for the older student.

Light. Because "... older adults are more sensitive to glare" [LOF, 215], the CRTs used by seniors should be equipped with glare-reducing filters. Additionally, it is important to avoid highly polished floors and walls; both the floors and walls could reflect too much light. [LOF, 215]. The training room walls, as well as the floor, could be carpeted to help in eliminating glare and absorbing extraneous sound.

The amount of light that the pupil allows into the eye diminishes as a person ages; an individual at age 80 needs three times as much light as a teenager. "Consequently, substantially greater lighting is needed in a classroom setting where older people are participating." [PET, 94] "The increased illumination should be diffused as much as possible to ensure reduced glare and should come from behind and above the older learner in order to optimize vision." [PET, 96] Peterson recommends desk lamps in some of the learning areas as a way of providing sufficient lighting on an individual basis. At each workstation, desk lamps could be used to illuminate any instructions or exercises that the senior student would need to consult; these might have dimmer switches to permit individual adjustment of the lighting.

Sound. "...teachers should make their presentations in a loud voice with as much precision in diction as possible." [PET, 100] Some of our students have complained that "because I have a hearing problem I need to sit in the front of the class;" however, there are so many who admit to being hard of hearing, it is impossible to seat them all near the front. As a result, the instructor needs to be aware of the reduced hearing level and speak loudly enough so that those who are in the back of the room can hear, yet make sure the words are pronounced clearly--even spelling out some of the words used and/or writing them on the board. (One student would bring a hearing aid with him, insert it before class, and remove it after class. A 'sound system' with individual speakers and volume controls at each workstation would be a possible way of addressing the hearing problem).

Use a lower tone of voice. Note: Peterson specifies tone or pitch, not volume; perhaps a male voice delivers information better to seniors. "...unvoiced consonants (s, c, t, p, g, for example)...are high-pitched and therefore difficult to hear... [women] with high-pitched 'squeaky' voices may prove impossible for the older students to understand." [PET, 100] Additional recommendations include avoidance of "s", "sch", and "f" sounds where possible and speaking slowly while facing the students (NOT the blackboard!). [LOF, 215] For the same reason the senior student may not be able to hear the high-pitched "beep" of the machine, usually indicating an error and the need for an action by the user. However, we have not encountered this as a problem - yet. Finally, participants should not converse among themselves--it is a distraction and interferes with the hearing of others. [PET, 101] (Also they tend to give each other the wrong information!)

Interference. There are several ways interference can occur.

> First, it can result from the conflict of previous, personal knowledge with the new knowledge to be learned. Second, two learning tasks undertaken at the same time can interfere with each other. Third, subsequent learning can interfere with the intended learning. [PET, 77]

Suggestions for handling these three types of interference follow:

> Interference #1 - Minimize conflicts between new and old knowledge [PET,

78], such as that experienced by individuals who have used Apple or Tandy equipment and are now using IBM or IBM-compatibles, or students switching among word processing programs.

Interference #2 - Stick to one task or topic; do not wander. To reduce the distraction of different topics, concentrate on one task at a time and let them learn it well first. [PET, 79]

Interference #3 - concentrate on the learning experiences sufficiently--give time to review, reflect, and apply them thoroughly to avoid forgetting.[PET, 79]

VI. TEXT

Organization of Material. (1) "Evidence is persuasive that older persons are less likely than others to spontaneously organize as a way to help memory."[PET, 81] Examples of helpful advance organizers the instructor can provide are course syllabi, class notes, summaries, and lists of informational items. (2) The content of the material should be presented in the same order in which the older learner is expected to apply it; transition to a different sequence is difficult. "The teacher must carefully choose the material to be presented in written form so that it closely conforms to the lecture presentation". "If the oral and the visual presentations are not similar, older students may experience the interference that occurs when they divide their attention." [PET, 88]

Hard copy materials handed out in class should be in LARGE, **DARK** PRINT. [YEO, 7] (Make sure that the ribbons in the printer are new enough to produce a dark print so the seniors can more easily see what they have generated).

VII. CLASSROOM LAYOUT

A recommended classroom layout might consist of two rows of four workstations (microcomputers on stands with desk space, lights, and possibly speaker units) facing an instructor table with workstation and overhead projector. A VCR and TV monitor should be placed at the back of the room, so that the seniors would turn around in their seats for video presentations. They should not take notes on the VCR sessions; we simply want them to absorb the information, without the distraction of taking notes. Any notes should be provided by the instructor, paralleling (exactly) the content of the video.

VIII. CONCLUSIONS AND SUMMARY

From our review of the literature and our experience in the presentation of introductory microcomputer courses for seniors at Florida Atlantic University, we have become aware of the special needs of such students, both physical and psychological. An understanding of the learning style of older adults is essential to avoid the frustration experienced by instructors accustomed to the comprehension rate and flexibility of younger students. Given an orientation to the needs of seniors, however, opening the world of microcomputers to them can be a stimulating and enjoyable experience. Careful attention to planning, scheduling, and development of training materials is essential if an introductory microcomputer course for persons over sixty-five is to succeed. Additional research is, of course, needed to determine such success criteria as the best delivery methodologies and the optimal sizing of completion units.

IX. REFERENCES

(1) [KEA] Kearsley, Greg and Furlong, Mary (editors). Computers for Kids Over Sixty. San Diego: Park Row Software, 1988.

(2) [KNW] Knowles, Malcolm S. The Modern Practice of Adult Education: From

Pedagogy to Andragogy. Chicago: Association Press/ Follett Publishing Company, 1980.

(3) [LOF] Loftus, Sheree Lee, review of videocassetes: "Aging: Sensory Losses" and "Speech and Language Problems of the Aged," The Gerontologist, 26 2 (Apr 1986), 214-216.

(4) [MUN] Munson, Lawrence S. How to Conduct Training Seminars. New York: McGraw-Hill Book Company, 1984.

(5) [PET] Peterson, David A. Facilitating Education for Older Learners. San Francisco: Jossey-Bass, Inc., 1983.

(6) [SAK] Sakata, Reiko and Fendt, Paul F. "Learning Capacity and the Older Adult: Implications for Lifelong Learning." Lifelong Learning: The Adult Years. IV 10 (Jun 1981), 10-13.

(7) Yeo, Gwen. "'Eldergogy' A Specialized Approach to Education for Elders". LifeLong Learning: The Adult Years. 5 2 (Jan 1982), 4-7.

Assessment of Noncognitive Computer Education Outcomes in Higher Education: the Case for the Lab Practical

Stanley Kowalski, Jr.
Marilyn K. Pelosi
David L. Russell
Western New England College

ABSTRACT

Recently, the AACSB has called for outcome assessment of business education in two areas: cognitive, or conceptual knowledge and noncognitive, or skill-oriented knowledge. While the need to assess skills encompasses the entire business curriculum, little attention has been paid to the assessment of skills in Information Systems education. This paper discusses one attempt to rectify this by use of the "lab practical", a technique borrowed from the physical sciences. This method is discussed in some detail, and the advantages and disadvantages of two implementations of the method are discussed. This is followed by two illustrative examples of the method in which the method is applied to two commonly-used software packages, Lotus 1-2-3 and dBase III Plus. A discussion of the advantages and the pitfalls of the methods follows.

An Empirical Determination of Computer Literacy

Elaine A. Crable
Phillip D. Jones
Xavier University

ABSTRACT

A review of the literature reveals multiple definitions of the term "computer literacy." This research extended the work of Tsay and Soloman to provide experimental evidence for a definition. Data for 239 structured interviews indicated that the importance of various factors to perceived computer literacy is highly job specific. Results are translated into a number of implications for trainers of computer skills.

An Empirical Study of the Computer Literacy Course at a Small College

Kathryn McCubbin
Christopher Newport College

ABSTRACT

There is general agreement in the academic community regarding the need for an elementary computer course in the baccalaureate curriculum. However, the content of such a course is controversial. The source of the controversy arises from what the faculty of the college perceive to be needs of their respective disciplines regarding computer science knowledge. In addition, students are exposed to computers in the general environment as well as in K-12 programs at an ever increasing rate. This raises the question of how frequently the content of an entry level computer course should be updated. This paper describes the results of faculty and student surveys conducted during both semesters of the 1988/89 academic year at a small southeastern college. The most interesting result of the surveys indicates that student exposure to computers in the general environment as well as in pre-college education are not an advantage when it comes to learning college level computer concepts.

THE ROLE OF THIRD GENERATION LANGUAGES PAST, PRESENT, AND FUTURE

Karen J. Ketler
Eastern Illinois University

Susan M. Moncada
Indiana State University

The role of the third generation languages in the work-place is in transition. IBM's new Systems Application Architecture (SAA) may be the catalyst for redesigning the data processing function in an organization. With SAA, it is predicted that the staffing of the IS function will focus more on personal computer development than mainframe development. The 4GLs may finally replace the 3GLs. This paper reviews the evolution of programming languages in an attempt to predict the future.

Since the delivery of the first commercial computer to the U.S. Census Bureau in June of 1951, developments in computer hardware and software have been phenomenal. Like any other industry, products marketed are largely determined by consumer demand. Some products remain prominent, while the significance of others wan. Over time, a shift in demand has also been experienced in terms of computer languages.

Questions continuously facing computer educators involve how best to prepare future information systems professionals. One of those questions is the changing role of the third generation languages in the work-place. It is in transition. Will the 3GLs be replaced by the 4GLs? What effect will IBM's new Systems Application Architecture (SAA) have on systems development and the importance of the 3GLs? Historians suggest that reviewing the past provides insight to the future.

THE PAST

Originally, the language of the computer was unique to the individual hardware being used. Vacuum tubes were switched on or off by using plug boards physically wired by computer technicians. During the early 1950's, the mechanics of switch setting was replaced by machine language programming. Programs were characterized by instructions written in binary. Combinations of 1's and 0's represented switches turned on or off. Machine language became known as the first generation computer languages.

The second generation languages soon evolved. Languages, such as assembler, were characterized by abbreviations and mnemonics. While the 2GLs greatly reduced the complexity of machine programming, they were less efficient from an execution viewpoint, because programs now required translation into machine language.

The third generation languages followed the assembly-language era. Languages, such as COBOL, FORTRAN, BASIC, and PL/1, became known as the high-level languages. They make use of English-like syntax and combined numerous machine or assembly commands into a single instruction.

During the late 1960's and early 1970's, the third generation languages started to flourish. Prior to that time, data processing departments were using 2GLs. The departments were centralized and

generally consisted of two areas: systems and applications programming.

Systems programming was and still is the functional area in the data processing department responsible for writing, debugging, and updating operating systems software, which controls hardware. It includes such components as file management, input/output controllers, and utility packages. These programmers evaluate the performance of the operating systems by monitoring and measuring the utilization and efficiency of the entire computer system. They trouble shoot, diagnose, and remedy software failures. They determine the need for new software, evaluate alternatives, and develop, acquire, and implement the selected software [3].

Application programmers, on the other hand, develop programs to solve problems or provide information to management. These programs include payroll, inventory, accounts payable, etc. While systems programmers are primarily concerned with the design, implementation and maintenance of programs that control hardware, applications programmers are merely users of the hardware.

Application programmers quickly adopted the 3GLs, because the 3GLs promised to increase programming productivity. They were and still are much easier to learn and use than the 2GLs. Each 3GL was standardized. As a result programs written in 3GLs are portable. This means that they can be used on most computer systems with little or no modification. The portability associated with the 3GLs gave rise to the prepackaging of software sold by third party vendors.

Systems programmers, on the other hand, continue to use the assembly language. The 3GLs require a complex compilation process. This results in the use of additional computer resources and a deterioration of hardware efficiency.

Although programmer productivity increased from the use of the 3GLs by application programmers, the number of requests for computer applications from users grew more rapidly. User dissatisfaction grew as the backlog of requests remained unfulfilled. The detailed nature and complexity of the 3GLs restricted their use to the professionally trained data processing professional. In order to reduce the need for programming services, decentralization, distributed data processing and the fourth generation languages (4GLs) evolved.

THE PRESENT

Distributed data processing is characterized by physically separated computers which are interconnected through communication facilities [2]. The control and responsibility of computer resources are distributed to the levels within the organization where they are needed by the end-user [10].

For the end-user to benefit from local computing facilities, a simpler programming tool was needed. The fourth generation languages (4GLs) were the answer. Developed intentionally for the end-user, they use the terms and phrases of the end-user's area rather than the technical jargon of the data processing department. Fourth generation languages are not procedurally-oriented. Instead they describe the desired results rather than the details of actions needed to achieve the results. They have query language capabilities for direct access to databases [2]. They are typically used for spontaneous, ad hoc reporting.

There have been several disadvantages associated with the 4GLs. Foremost is the fact that many users have had little formal training in the development of computer applications. Errors often go undetected, because extensive testing and debugging is not performed.

Industry experts estimate that one of every three electronic spreadsheets contain errors [1]. Secondly, users are sometimes spending more time learning how to develop applications rather than performing their basic assignment of duties. Experts estimate it takes a new end-user 24-60 hours of keyboard time just to learn how to use a spreadsheet program. It takes the average mid-level manager five hours of practice and review time per week for approximately six months to make the spreadsheet a viable business tool [6]. In addition, program duplication often exists, because different departments are not aware of the efforts of other departments. Finally, the 4GLs are not standardized. As a result, they are not portable. Prepackaged systems are difficult to obtain.

To reduce some of these problems, the data processing programming shop changed. In addition to systems and applications programming areas, a user-oriented area was created. Its purpose has been and continues to be aimed at helping end-users solve problems utilizing computer technology. This area usually performs three functions. It strives to keep abreast of research and development in both hardware and software technologies. Second, it sets organizational standards and procedures to guide applications development and governs the purchase and maintenance of hardware and software. The third, and perhaps the most important function, is to provide consulting and training services for end-users [5].

While the systems programmers use the 2GLs and the application programmers use the 3GLs, the user-oriented area of the data processing function, in addition to the end-users, use the 4GLs. Currently each language generation has found a niche in the organization. Third generation languages are primarily being used for administrative applications that affect the entire organization. They are generally used with a structured problem by the application programmer. Fourth generation languages are typically used by end-users who need spontaneous answers to "what if" questions. The applications developed tend to be oriented for a single department and have a smaller focus.

THE FUTURE

Problems of nonstandardization continue to plague user departments. This nonstandardization has also caused problems for IBM. Due to the lack of standardization among its three major models: (1) the 370's, (2) the systems 3X's (34, 36, and 38s), and (3) the personal computers, IBM has recently lost ground to Digital Equipment Corporation's VAX machines. In retaliation, IBM has made plans to standardize, not only its hardware, but also the software that the machines use. What may determine the future of the 3GLs and the 4GLs may be computer hardware--in particular, IBM's Systems Application Architecture (SAA). SAA is a framework for the creation and distribution of common applications across networks connecting the systems 370's, 3x's, and PC hardware. SAA is IBM's promise of software portability, consistency, and standardization.

SAA imposes layers of order over the incompatibilities that exist among operating systems, programming tools, languages, and communications protocols [4]. With SAA, applications developed using supported languages and tools will be transportable between the PC, mid-sized systems, and the large mainframe systems. Although SAA will allow users to continue developing applications using previously established languages and tools, these applications will not be transportable. As a result, SAA may be the catalyst for redesigning the data processing function in an organization.

SAA originally was designed to support:

(1) A software interface that strongly

resembles Microsoft Windows.

(2) SQL as the database query language.

(3) QMF as the report writer.

(4) CSP (cross system product) as the 4GL.

(5) COBOL, FORTRAN, and C as the 3GLs.

(6) REXX for the job scheduling language.

In addition, IBM plans to include some 5GL artificial intelligence tools.

Typically, as users adopt new software packages, they must repeatedly learn to use new commands and screen layouts. SAA's Common User Access establishes a single, standard interface for all software packages. The interface is similar to Microsoft Windows. With this interface, special function keys will have similar meanings in all software packages. One of the advantages associated with this feature is a reduction in user retraining costs.

SAA supports IBM's Structured Query Language (SQL) for querying databases and Query Management Facility (QMF) for report writing. SQL has widely been used on personal computers, and is currently being adapted for IBM's mid-sized systems (3x's). Inconsistencies between SQL and DB2 on the mainframes still need to be resolved [4].

CSP is the 4GL supported by SAA. CSP permits easy migration between IBM's various hardware environments. It was designed primarily for the mid-sized systems. Although it has been available for use for almost a decade, it is not widely used. In a survey of large systems users by International Data Corporation, only 45% of the users had programmed with a 4GL, and only 13% had used CSP. Users of CSP indicated that it was primarily used for prototyping and not as a production tool. CSP apparently lacks the robustness and performance needed by large systems users. In addition, no common library for storing and accessing source code statements is available [4].

Other problems exist. CSP has no means by which to incorporate artificial intelligence concepts. It is efficient for small projects, but larger developmental efforts must be broken into smaller segments. This not only requires design modifications, but results in additional programming expenditures. Michael Graham of World Savings has been quoted as saying: "It's a little better than a 3GL if you're working on a small project with junior to mid-level programmers. But if you have people with a lot of experience, COBOL can actually be faster" [7].

The standardized 3GLs included on SAA are COBOL, C, and FORTRAN. Users became upset that RPG and CICS would not be supported by SAA. The fear of software obsolescence annoyed many IBM users. After reconsideration, these languages and tools will also be available on SAA. CICS, however, will be limited to IBM 370 users [8]. Although IBM says it will continue to support certain existing products that do not fit into the SAA architecture, the future of products like PL/1 may be tenuous.

REXX is the standard language being supported for issuing commands that invoke programs and systems facilities. This software package was created in 1983 for the VM/CMS mainframe systems. It is a general purpose, procedural language that is used as both a programming and job scheduling language.

SAA's programming and data access interfaces will allow end-users to develop reporting and analysis applications. Database tuning and

applications optimization will be needed less often and require less technical proficiency to implement. In addition, large programming staffs and technical specialists may no longer be necessary. Lewis Priven, IBM's director of SAA, predicts that end-users will do more of the applications work. Peat, Marwick, Main and Company has already combined 50 mainframe developers and 20 office systems developers into a single group. They predict that the staffing ratio of mainframe developers to the end-user personal computer developers will invert, because it is easier to develop applications for a personal computer [4]. As the demand for personal computer developers increases, the importance of 3GLs will decrease.

DMR Group, Inc. of Montreal predicts that the line between personal computer and mainframe development will disappear. There will eventually be one development group working on all IBM machines ranging from the 370 mainframes to the personal computers. Technology will no longer be the controlling force in data processing. It will be the application requirements and those who understand business that will determine the programming languages used. Centralized IS organizations will shrink as programming duties are assigned to business units. In addition, the centralized IS organizations will be viewed as a utility function within the corporation [4].

Will IBM be able to deliver its promises? What effect will SAA have on IS staffing and structure? Will the 4GL (CSP) be able to handle the large applications? When applications can be developed on personal computers and then transported automatically to a mainframe, there will be less demand for the traditional mainframe staff. Less demand for the traditional mainframe staff may equate to less use of the 3GLs. The 1990's may very well be the decade of the 4GLs.

BIBLIOGRAPHY

1. Creeth, Richard. "Microcomputer Spreadsheets: Their Uses and Abuses." Journal of Accountancy. June, 1985, pp. 90-93.

2. Davis, Gordon B. and Olson, Margrethe H. Management Information Systems. New York: McGraw-Hill, 1985.

3. Eastern Illinois University Job Description, Charleston, Il.

4. Kull, David. "SAA: Master Plan or Grand Illusion?" Computer & Communication Decisions. June, 1987, pp. 84-94.

5. Kroenke, David. Management Information Systems. Santa Cruz, CA: Mitchell Publishing, Inc., 1989.

6. Middleton, Bill. "Ten Tips for Speeding Up Spreadsheet Learning." Computers in Accounting. March/April, 1986, pp. 17-19.

7. Moad, Jeff. "CSP: The Weak Link in SAA." Datamation. November, 15, 1988, p. 49, 52.

8. Moad, Jeff and McWilliams, Gary. "SAA: The Yellow Brick Road to Cooperative Processing." Datamation. July 1, 1988, pp. 38-48.

9. Moran, Robert. "IBM Completes Its Big Picture." Computer Decisions. September, 1988, pp. 58-62.

10. Wolff, Robert L. & Litecky, Charles R. "Distributed Data Processing Controls." The EDP Auditor. Fall, 1982, pp. 1-14.

Expert Systems, SQL and 4GLS: Flexibility and Power

John C. Shepherd
Duquesne University

ABSTRACT

Instead of using an expert system shell or an artificial intelligence language, such as PROLOG or LISP, developers can use SQL to construct certain kinds of rule-based expert systems. This paper shows how SQL and a 4GL were used to develop a large expert system to monitor drug usage patterns for hospitals and large insurance companies.

INTRODUCTION

Most expert systems are characterized by the use of an AI-type language or an expert systems shell, through which rules are developed based on results derived from a knowledge engineering phase. Then data is imported from the organization's existing databases or files, or the data are manually keyed.

The rules attempt to embody the expert's knowledge as a collection of "IF-THEN-ELSE" statements, which if "true" are said to have been fired. Elegance and sophistication can be added in the form of probability statements concerning the degree of "truth," and the expected values of outcomes can be determined. If some of the rules are mutually exclusive, and if we can somehow embody this fact in the expert system, subgoals can be modeled.

The input data to an expert system varies from run to run and consists of the attributes of the object being evaluated by the expert system. In our implementation, we were evaluating the quality of prescriptions. Among the attributes of a prescription are the patient number and name, the pharmacy name, the physician name and code, the drug, its strength, how many day of the drug were prescribed and so on.

In many cases, the amount of this data can be staggering. In our model, the potential exists for millions of records to be processed against more than 300 rules. A large insurance company could have millions of customers, each having 5-10 prescriptions per year. The usual way for transferring such data would be to "unload" the data from the organizational database, then import it into the expert system shell. Clearly this is impractical.

Furthermore, oftentimes the end-user needs the ability to search the knowledge base for information not provided by the output of the expert system. For our purposes, once we determine that a particular physician is overprescribing a certain drug, or that a pharmacy is refilling a prescription too frequently, the experts want the ability to print ad hoc reports in varying levels of detail. This cannot be done with expert system shells. It requires a relational database. For us it was clear that an alternative to an expert system shell was needed.

This paper describes how we are using SQL and Informix-4GL to build a large expert system that monitors drug utilization for large insurance carriers and hospitals. We begin by describing the drug monitoring process.

Drug Monitors

Drug monitoring is the process of examining how drugs are used by the three participants in the drug-use cycle: physicians, pharmacies and

patients. Some of the possible areas of study for insurance companies include over and underutilization, fraud and abuse, failure to substitute a generic drug, and duplicate therapy.

For hospital settings, drug interactions may result in morbidity and mortality and increase a patient's length of stay. Other areas of concern include over and under dosing and adverse drug reactions (ADRs).

Overutilization/Fraud and Abuse

Overutilization occurs when a patient exceeds the recommended frequency of dosing for a given drug over a period of time. We define fraud and abuse to be: a patient shops from pharmacy to pharmacy, getting the same prescription filled each time; the patient goes from physician to physician, getting multiple prescriptions, or a combination of both. It can also occur when a pharmacy bills for prescriptions not dispensed.

As an example, an easily abused tranquilizer such as Valium, is prescribed three times a day, but the patient takes it more frequently; in a 30 day period of time he, or she, consumes 60 days worth of the drug. This can occur because the pharmacy ignores the refill pattern or the patient is a "shopper," visiting multiple pharmacies and/or physicians.

The algorithm for calculating overutilization is:

- Determine the Actual Duration of Therapy (ADT).
- Determine Excess=Total Days Supply-ADT
- Calculate OU = Excess/ADT.

Theoretically, if OU is over zero, we have overutilization. Instead of using 0.00 we used a figure of .25 as the indicator of overutilization. In the above example, Excess would be (60-30), ADT would be 30, and OU would equal 30/30 or 1.00, meaning that the patient consumed 100% over the amount prescribed.

Underutilization

This occurs when patients are noncompliant, that is they skip several doses. For example, in a 30 day period, the patient only consumes a 15 day supply. Such a pattern is frequently seen with geriatric patients and may occur because of drug side-effects, forgetfulness, etc.

Underutilization is determined in the same manner as was overutilization. However, a ratio of at least -.25 was used. In the example above, the ratio would be: (15-30)/30 = -.5.

Failure to Substitute a Generic Drug

In many cases, a less expensive generic drug can be prescribed rather than the more costly brand name drug. Repeated for thousands or millions of patients, this cost can be substantial.

One of our tables is NDC, which contains a row for most of the 100,000 available drugs. One of its columns is NDC_GENERIC, which represents the NDC# of the least expensive generic substitution. We in effect treat this column as a foreign key, which instead of referring to a parent table, refers to itself. Therefore, we are able to look up the cost of the least expensive generic equivalent.

The cost of the failure to substitute a generic, is the excessive per unit cost of the brand name over the generic cost, times the number of units used by the patient.

Duplicate Therapy

This would most likely occur when a patient receives two drugs having the same generic description from two different physicians and/or pharmacies. Costs associated with it include the cost of the redundant drug, possible

reactions to the over dosing, which leads to office visits or hospitalization, and, for some drugs, possible addiction.

Here we had to look for the possibility that two concurrent prescriptions specified two different brand names, but the chemical content was the same.

Let's now look at some the hospital-related drug monitors.

Drug Interactions

This occurs when two or more drugs administered concurrently, produce an adverse effect. Although there are thousands of such interactions, there are only several hundred that are clinically significant, and for which we developed rules.

Here, the system has to be able to handle temporal relationships: we need to know if a patient is on two drugs at the same time. Also, we had to determine the time increment (onset) between the start times of drug B and drug A.

The severity of the interaction is usually a function of the duration of the therapy, the age of the patient, and the daily dose. For example, the severity of the interaction between Tagamet (cimetidine) and Theodur (theophylline) is an increasing function of the patient's age and the daily dose. That is, patients over 65 with a daily dose of Tagamet over 800mg and on more than 900 mg of Theodur for more than two weeks are at much greater risk than a younger patient on smaller doses for a shorter period of time.

Other factors that contribute to the severity of the interaction include the change in dosages over time, the length of time the patient has been on the second drug, and the strength of the second one.

More than 200 potential drug/drug interactions were defined. As shown in a later section, we defined a RULE table that has columns for the parameters defined above. For example, there are columns for drug A, drug B, age, dosage of drug A, dosage of drug B, and so on. This table is the essence of our model.

For some of the interactions, a single rule was postulated, but in those cases where the severity varied by age and dose, many rules had to be formulated for each interaction.

Dosages Over or Under a Recommended Amount

Here, the experts defined a recommended maximum dosing level for those drugs that can result in toxicity, adverse reactions, or addiction. We had to calculate the average daily dose for each drug taken by a patient, over an extended period of time.

Determining the average daily dose for a patient requires the following calculation: the total milligrams dispensed divided by the total day's supply.

What To Do

After we decided to automate the process, we had to choose a language. We considered using an expert system shell but decided against it because of the need to be able to use the power of a relational database, so we could go back to the detailed data and select prescriptions based on patient, physician, pharmacy, and so on. To have both the expert system shell and the relational database, we would have had to convert millions of records from the RDBMS to the shell and vice versa. We elected to use SQL and Informix-4GL instead.

EXPERT SYSTEMS

A rule-based expert system contains a knowledge base and an inference engine. The inference engine is a program that contains logic to make inferences based

on the knowledge base, which contains a set of rules, usually expressed as some sort of "IF-THEN-ELSE" statements. The two most common inference methods are backward and forward chaining.

For our application, we needed to use forward chaining because we wanted to screen input prescription data and have the system reach conclusions consistent with rules postulated by the experts.

The data used by the inference engine can be primitive (stored in the knowledge base) or derived. The derived items can be computed via a simple function or based on logical inferences. As an example of a derived item based on an inference, suppose that we have a rule that says:

```
IF a Physician's Average Cost > the
Average Cost of All Physicians
  THEN
    Display the Physician's Average
      Expected Cost
END-IF
```

The SQL code would be:

```
        SELECT MID, AVG(COST)
            FROM ORDER
              GROUP BY MID
                 HAVING AVG(COST) >
                   (SELECT AVG(COST)
                       FROM ORDER))
```

The SELECT statement says to group the physicians by code, but to include only those physicians whose average cost exceeds the overall average. In effect, we have embedded the rule into the SELECT statement.

Through the power of SQL we can also emulate the forward chaining. Consider the following example.

```
IF a Patient is on 'triazolam'
  THEN
    IF the Dose of 'triazolam' > .25mg
       THEN
         IF the Patient is over 64
           THEN
     There is a .25 probability of a $ 65
     office visit resulting from mental
     confusion and a .1 probability of a
     $2,000 hospitalization
          END-IF
     END-IF
END-IF
```

Here we have 3 (nontemporal) rules, which are fired in the forward direction in an attempt to determine the expected cost of a particular (inappropriate) drug regimen.

Assume we have a SQL table called ORDER, which contains all the prescriptions for all patients. To accomplish the above using SQL, we would simply code:

```
SELECT PATIENT.PNAME, "The cost associated
   with mental confusion is",
   .25 * 65 + .1 * 2000
 FROM ORDER,PATIENT
   WHERE (ORDER.GENERIC='triazolam' AND
       ORDER.STRENGTH > .25          AND
       TODAY-PATIENT.BDATE >64       AND
          PATIENT.PID = ORDER.PID);
```

Until the advent of SQL, temporal (time-related) relationships were almost impossible to manage. They required complex array processing or the continual opening and closing of files to make multiple passes. You can see this in the next section, which discusses drug interactions.

SQL As An Expert System Shell For Drug Monitoring

Before we look at the model, we need to describe the major components of the database. Because of nondisclosure agreements between the author and the firm for which the system was developed, many items aren't shown.

The Database

The major components of the database include:

```
ORDER (PID, NDC#, STARTDATE, REFDATE,
    DRUG, DAYS, UNITS, CUM_DAYS,
    STRENGTH, QTY, RX, MID)
```

Where:
PID is the patient's identification number,

NDC#, the NDC number of the drug,
STARTDATE the date that the drug was begun,
REFDATE the most recent refill date,
DRUG, the drug name (generic),
DAYS, the most recent days supply,
UNITS, the number of metric units dispensed,
CUM_DAYS, the cumulative days supply,
STRENGTH, the level (amount) of the drug,
QTY, the quantity dispensed,
DC_DATE, the calculated discontinue date
(STARTDATE + CUM_DAYS),
RX, the pharmacy's identification number,
MID, the physician's state license number,
NDC(NDC#, BRAND, GENERIC, MAX_REC_DOSE,
MIN_REC_DOSE, REC_MAX_DUR, AWP,
NDC_GENERIC)

Where:

BRAND, is the drug's brand name,
GENERIC, the generic description,
MAX_REC_DOSE, the recommended maximum
dose,
MIN_REC_DOSE, the recommended minimum
dose,
REC_MAX_DUR, the maximum recommended
duration of therapy,
AWP, the Average wholesale price of the drug,
NDC_GENERIC, the NDC# of the least cost
generic substitution
PATIENT(PID, PNAME, BDATE)
RULE(RULE_NO, NDC_DRUGA, NDC_DRUGB,
ONSET, LL_AGE, UL_AGE, CHG_DRUGA,
CHG_DRUGB, DOSAGE_A, DOSAGE_B,
EXP_COST1, EXP_COST2)

Where:

NDC_DRUGA is the NDC # of the first drug,
NDC_DRUGB, the NDC of the added (second)
drug,
ONSET, the elapsed time between the start
dates of the two drugs,
LL_AGE, the lower limit on patient age,
UL_AGE, the upper limit on age,
CHG_DRUGA, the change in strength of the
first drug,
CHG_DRUGB, the change in strength of the
second drug,
EXP_COST1, the expected cost of an office
visit if this "rule" fires,
EXP_COST2, the expected cost of
hospitalization, if this rule fires.

The RULE table is the main one that we use in our expert system, and is, essentially, our knowledge base. It is used to test for drug interactions and adverse drug reactions.

Note that the ORDER table isn't in 3NF because we have a computed field (DC_DATE). This was intentionally done to decrease the run-time of the expert system. Instead of computing the DC_DATE as a virtual field, we carry it in the table. Also, we carry the GENERIC description in the table (which means a nonkey determines another nonkey) for the same reason: performance.

The first area we explore is drug interactions. Remember that there are several parameters to consider: patient age, dosing levels, length of therapy and so on.

Drug Interactions

We don't have the space to look at all the drug monitors that were developed. Accordingly, we shall only examine SQL code for drug interactions, over/under recommended levels, and over/under utilization.

As an example of a drug interaction that we handled, consider this pseudo-code:

```
IF a Patient is on 'cimetidine'
  THEN
    IF the Patient is also on 'theophylline'
       THEN
         IF The Patient started 'theophylline'
          at least 7 Days after 'cimetidine'
          THEN
            IF the dose of 'theophylline' > 500mg
              THEN
               IF the Patient is over 65
                THEN
                There is a potential interaction of
                 Level 8.
                    END-IF
               END-IF
         END-IF
    END-IF
END-IF
```

Because of SQL's set-processing capabilities, we could accomplish the above by using an alias with a join:

```
SELECT PATIENT.PNAME, "Severity level = 8"
 FROM ORDER A, ORDER B
  WHERE A.PID = B.PID                    AND
   A.PID = PATIENT.PID                   AND
   A.GENERIC = 'cimetidine'              AND
   B.GENERIC = 'theophylline'            AND
    B.STARTDATE - A.STARTDATE > 7        AND
     B.STRENGTH > 900                    AND
     A.STRENGTH > 800                    AND
     TODAY - PATIENT.BDATE > 65
```

Each possible interaction, and variation was handled similarly.

In fact, we created a dynamic SQL SELECT statement within Informix's 4GL, that used cursors to process each prescription against all the rows in the RULE table. By coding in mutual exclusivity, we were able to implement subgoals, thereby skipping over rules that were superfluous.

Handling Prescriptions Over/Under a Recommended Limit

Our code for determining if a patient is prescribed more than the recommended dose looks something like this:

```
SELECT PID,ORDER.GENERIC,
 ORDER.STRENGTH*UNITS/DAYS AMT
    FROM ORDER, NDC
    WHERE ORDER.NDC# = NDC.NDC# AND
        AMT > NDC.REC_MAX
```

For example, assume the recommended maximum recommended daily amount (REC_MAX) for NDC# 123 is 200 mg. A prescription calls for 90 units of the drug (UNITS) at a strength of 100mg. A 30 day supply (DAYS) is specified. The actual daily dose would be: 90 * 100 /30 or 300mg per day. Since this exceeds the recommended daily maximum, our system would flag this prescription.

Under/Over Utilization

Recall that our rule specified to examine prescriptions longer than 60 days, where the patient consumed more than 25% over what was prescribed. The SQL code for calculating overutilization appears below.

```
SELECT PID,GENERIC, (CUM_DAYS-(DC_DATE-
 STARTDATE)/(DC_DATE-STARTDATE)) OVR
    FROM ORDER
    WHERE (DC_DATE-STARTDATE > 60 AND
     OVR > .25;
```

Assume a prescription that has been "active" for 60 days. For example, the DC_DATE is 03/01/90, and the STARTDATE value is 01/01/90. The actual days supply of the prescription is 90 days. Then OVR would be: (90 - 60)/60, or .50 - the patient used 50% more of the drug than he, or she, was prescribed.

CONCLUSION

We feel we are breaking new ground by using SQL to implement an expert system for tracking inappropriate drug use. Because of the built-in SQL facility for handling temporal relationships, we were able to overcome what had previously been labeled as impossible.

We demonstrated how to use SQL to construct a rule-based expert system with forward chaining. Also discussed was how to construct logic-based derived values and how to manage the temporal relationships.

REFERENCES

1. Chandler, J.S., and Liang, T.P. Developing Expert Systems for Business Applications, Merril Publishing Co., 1990.

2. Cohen, B., "Merging Expert Systems and Databases,"AI Expert, February, 1989.

3. Holsapple, C. and Whinston, A.B., Business Expert Systems, Irwin, 1987.

FOURTH GENERATION LANGUAGES: THEIR IMPACT ON INFORMATION SYSTEM DEVELOPMENT TODAY AND TOMORROW

Herman P. Hoplin
School of Management
Syracuse University

Kimberly A. Free
J.T.S. Computer Services, Inc.
Rochester, New York

ABSTRACT

Educators concerned with supporting the changing workplace have become more and more interested with the connectivity problem of plugging in Fourth Generation Languages (4GLs) to work on business applications. As time passes, the academic tendency to hang on to procedural languages has caused an environmental barrier which has become an opiate between academe and the private sector. 4GLs not only bridge a generation gap between the two environments but remove much of the mystery from productivity problems associated with programming and with user demand for timely support in the solving of business problems. This paper looks into this trendy alternative for proactive innovations.

INTRODUCTION

Pick up a computer magazine and odds are there will be an article or an advertisement on 4GLs. Is this the industry's newest fad or its latest technology? The paper addresses 4GLs and their impact on information systems development today and in the future. It begins by defining the term 4GL. A comparison of the strengths and weaknesses of 4GLs and traditional programming languages (TPL) will be presented followed by the forces behind the transition to 4GLs. How traditional Management Information Systems (MIS) departments will be affected by this transition also will be discussed.

WHAT IS A 4GL?

The term 'fourth generation language' has captured the software industry's attention. It is also one that carries a lot of confusion as to what it means. Ambiguous terms are all too common in the software environment. What's unusual about 4GLs is that they are in great demand and heralded as the miracle cure for ills such as low programmer productivity, applications development backlog, and timely action on ad hoc reports.

While this may be an exaggeration in some companies, MIS departments are being barraged by advertisements for an unending variety of products claiming to be 4GLs. Many developers are becoming overwhelmed by the possibilities and want to know how one determines if a product truly is a 4GL. The opinion of many authorities in the field, one that we share, is that a 4GL must be: powerful and flexible so that new applications can be developed fast; non-procedural--users tell the language what to do rather than *how* to do it; easy to maintain and enhance; capable of supporting the rapid prototyping of systems quickly; able to promote the concept of 'visual thinking'--what you see is what you get; and data independent.

The fact that a group of 4GL experts can agree on the characteristics of 4GLs, even if they can't agree on specific wording, indicates the strength of the concept. They agree that a 4GL should be used to: (1) Improve programmer productivity, (2) Obtain better information system requirements faster, and (3) Facilitate end user computing.

James Martin lends credibility to 4GL tools by stating that they "provide mankind with a leap forward in the ability to put the computer to use."[1] To make the term meaningful on what a 4GL is, one must look at what a 4GL should do and how it should be used, not at specific products. To do this, a questionnaire was sent to 47 software developers from many environments. The purpose of the survey was to find out how these individuals define and view 4GLs. The results closely parallel the earlier consensus. Developers agreed on what a 4GL should do and be used for; however, little consensus was reached on examples. The respondents indicated that 4GLs should include the following characteristics:

- o Easy to learn, use, and maintain.
- o Employ menu driven user interfaces and offer built-in functionality.
- o Accomplish difficult tasks with ease.
- o Require little technical detail to program a task.
- o User tells the program what to do, not how to do it.
- o Utilize English-like language.
- o Can be used for prototyping.
- o Data independence.

The survey respondents also were asked to indicate how they felt about the expert's claim that 4GLs: (1) Improve programmer productivity, (2) Allow users a more hands-on role in system development, and (3) Allow systems requirements to be determined faster and more accurately. There was strong consensus among those surveyed that the first two points were benefits from using 4GLs. However, approximately half of the respondents disagreed with the experts' claim that 4GLs allow system requirements to be determined faster and more accurately.

The survey results reinforced our belief that 4GLs are a valid concept--one that focuses on producing user-friendly software quickly and easily. Our experience indicates that a 4GL is a unified set of tools that allows application developers to interface with other software in a manner that facilitates:

(1) Creation and maintenance of user-friendly application interfaces (menus and screens).
(2) Generation of formatted reports.
(3) Performing complex data queries.
(4) Accessing the database using a TPL such as C for real-time system tasks.
(5) Prototyping systems.
(6) Incorporation of the concept of visual thinking.

COMPARISON OF 4GLS TO TPLS

In the first comparison we look at the creation and maintenance of user-friendly interfaces within an application. ORACLE will be the 4GL. The entry screen has an order block with a customer ID, and an items block with product IDs. The F1 key takes you from the Order block to the items block. To implement this screen in ORACLE, SQL*Forms[2] is used as a visual forms editor. This tool allows the user to place text and data fields on the screen where they are to appear. Scrolling data regions occur automatically when you request multiple data records to be shown. Attributes such as mandatory fields and field justification can be chosen from a menu at each data field. The form has easily changed default values for the next field a user should go to. The tasks are done using a windowing menu system that lists the options that can be done at each step, making the tool easy to use (see Figure 1). The form takes care of retrieving data from or updating data to the database.

Implementing this same screen using a TPL would take significantly longer. One reason is that the programmer must state explicitly where each text and data field should appear on the screen. The source code must be compiled and run. Iterations of editing, compiling, and running continue until the form looks as it should. A second reason for the longer development time is that many functions which are provided to the applications developer by ORACLE must be written by the TPL programmer for each field on the screen. The F1 key functionality can no longer be implemented at the form level, rather this capability will need to be coded for each field.

From this comparison, the 'what you see is what you get' capabilities of 4GLs makes using them to develop screens a lot faster and easier than using a TPL. Conversely, using TPLs to create screens is time consuming and requires the services of a skilled programmer.

The second comparison involves complex, multiple table data queries using Informix-4GL. The requirement of the query is to retrieve all orders and the associated customer numbers which were active during the period. The data is stored in two files--order, which has information on orders and status, which contains the status codes associated with every order. A table of file layouts, sample data, and query results from this. Using Informix-4GL, the applications developer would use SQL to query the database. Since this is a data independent language, the user will only need to know the table and column names of the data needed by the query--they do not have to be concerned about the

format of the data. A single statement will return all rows of data meeting the selection criteria without the application developer telling SQL how to get the data. The capability of using the results of one query to act as a WHERE clause of another query makes SQL especially powerful as a querying tool.

Doing the same query using COBOL, for example, would require a compiled program rather than a single statement. Since TPL is a data dependent language, a programmer would need to know the layout of the two files, including the names and formats of every field in the file. The programmer would also need to describe the data files and tell the program how to retrieve the data row by row. On the other hand, SQL's biggest strength is the fact that it is data independent. If either table is changed, the SQL statement would still run as it is as long as none of the columns in the statement changed. The TPL program would need to be recompiled if any change occurred to either file--even if the change had no direct impact on the query.

It will be shown later, based on the differences presented in these comparisons, that prototyping is a useful tool particularly with 4GLs. Since it takes a significantly longer time to develop and modify applications using TPLs, it is difficult to justify prototyping, especially when it may be discarded after system functionality has been determined.

The two comparisons also demonstrate the advantages that 4GLs have in implementing the concept of visual thinking. It is more efficient to create software that needs to look visually pleasing and correct if you are viewing the screen the way it will look to the user when it is running. Tables 1 and 2 on 4GLs and TPLs, summarize the advantages and disadvantages of each.

After looking at the differences between 4GL and TPL, we offer the following projection. In the future, most interactive applications will be developed using a combination of 4GLs and traditional programming languages. The majority of an application (about 90%) will be written using a 4GL because programmer productivity is at a premium.

FORCES BEHIND THE TRANSITION FROM TPLS TO 4GLS

The primary forces behind the transition from traditional programming to 4GLs are management requirements rather than technological needs. Managerial concerns fall into two areas: controlling escalating systems development costs and implementing crucial business systems on a timely basis. These items go hand-in-hand and 4GLs address them in a number of ways such as: (1) Increased programmer productivity because the application developer only has to tell the 4GL what he/she wants done, (2) 4GLs allow for direct user involvement via prototyping, (3) 4GLs contribute to lower system development costs and quicker implementation, and (4) Greater portability.

INTEGRATING 4GLS IN THE MIS DEPARTMENT

MIS departments that choose to move from TPL to 4GL based systems will face major changes. The reason for this is that 4GLs challenge the procedures these departments have been using. In addition, the use of 4GLs opens up the design and implementation of information systems. The major change in organizations which move to using 4GLs is the software development process itself. Systems developed using TPL are typically implemented using a structured systems design approach. This approach entails writing volumes of specifications, design details, and test plans. These are reviewed by users via regularly scheduled walkthroughs. The process is usually instituted in such a way that one step must be completed before the next can begin, thus contributing to the time needed to implement a system using TPLs.

Contrast this to one of the more popular software development 4GL methodologies using prototyping. The process used to create this sample application, the prototype, is an iterative one where requirements are stated, partially implemented, and reviewed by the users. After this review, the requirements may be modified, the changes implemented and another review performed by the user (see Figure 2). The key, critical difference between this approach and the structured one is that a step need not be 100%

completed before the next one is begun.

Prototyping has one big advantage that the users get to see what their system will look like early in the process. There are four other important advantages: (1) Trims the time and paperwork involved in systems design, (2) Emphasizes interactive user involvement, (3) Provides a systems definition format that users can understand, (4) Results in a tangible product that can be used in the final system.[4]

Coupled with 4GLs ability to quickly implement system requirements, prototyping is a feasible and cost effective alternative to the formal development process that many MIS departments use today. Over two-thirds of the 4GLs survey respondents use prototyping as part of their software development process and 90% of these use the prototype as the cornerstone of new applications.

What impact will this change in methodologies have on MIS department managers, staff members, and system users when they transition from a TPL to a 4GL based system? The largest impact on MIS managers will be the manner in which they manage software development projects. As MIS come to play a more critical role in an organization which will happen as development costs and timetables decrease, MIS managers will need to interact more directly and frequently with other corporate departments. Traditionally, users and developers view applications development from different viewpoints. Getting these two groups to work together effectively can be a challenging task for the best MIS manager.

Transitioning from TPL to 4GL based systems will have the most impact on the staff members who will be responsible for developing software in the new environment. They will need to adjust to working with developers who have not worked with 4GLs. before. In addition, managers will expect applications developers to be more productive and to meet shorter development times than in the past. The survey results indicate that many software developers are already using what they consider to be 4GLs and are relatively happy with the results they are obtaining. These same staff members are aware of the pitfalls of using 4GLs and tend to be wary of accepting the claims of many commercially offered products. One survey respondent said it best what many indicated: "In general, the software works. Because it is 'young' software there are rough edges and many new releases to come. You can get some things done in a 4GL faster than a traditional programming language, but there are other things that just plain cannot be done in a 4GL and one must revert to the old standbys."

The last group to be affected by the transition from TPL to 4GL based systems is the users. They will now be expected to have a more active, hands-on role in applications development from requirements specification to system testing. This will give them a feel of greater ownership of the system they are using. It will also place a greater responsibility for the correctness of the system on them. Traditionally, users have not been involved in systems development until the end and problems with system functionality have been attributed to the MIS departments lack of understanding about the business process. Involving users earlier in the development life cycle should ensure better quality MIS.

From the foregoing, it can be surmized that transitioning from TPL to 4GL based MIS will have a dramatic affect on the typical MIS department. While there may be periods of upheaval and discontent, eventually MIS departments will realize the benefits of using 4GLs and related technology.

FUTURE DIRECTIONS OF 4GLS

Thirty years ago, arithmetic and logical operations were the fundamental programming algorithms. Today, editing, cursor movement, and help operations are considered to be even more fundamental programming tools. In the new era of personal computing, where will today's 4GLs lead to in programming technology?

Future 4GLs have their foundations in the 4GL products available today. Just as today's 4GLs are user-friendly, data independent and contribute to application developer productivity, so will tomorrow's. In addition to the features available today, the 4GLs of the future will include:

- Multi-windowing capabilities.
- Operating system features.

- o Additional productivity improvements.
- o Interfaces to object-oriented as well as relational databases.
- o Graphics capabilities (both as output and as tools for the developer).
- o Integrated spreadsheet and word processing capabilities.

These new features indicate a trend toward integration of an organization's information and access methods using a single product. They also show that as programming languages gain more functionality, application developers are changing their view of what a programming language is. For instance, few software professionals would have considered spreadsheets as a programming language five years ago.

In the interest of continuing the drive towards programming languages that are implementable by non-professionals, 4GLs of the future will continue to be non-procedural thus allowing the applications developer to focus on the results of the system rather than the details of its implementation. This drive will contribute to the next (5th) generation of languages, which will offer functionality such as repositories, artificial intelligence and knowledge banks.

SUMMARY

The intent of this paper was to look at 4GLs and their impact on the support role for corporate information systems development today and in the future. This was accomplished by reviewing how experts defined a 4GL and comparing this with the definitions provided by software developers via a survey and our own definition of what a 4GL is. It was shown that in order to define a 4GL, one must step away from specific products and look at the characteristics a 4GL must have such as quick application development, easy maintenance and enhancement, data independence, structure for ease of use, prototyping, on-line operations, and readily can be used effectively by non-MIS/DP users.

After establishing what would be considered a 4GL, TPLs and 4GLs were compared. It was shown that 4GLs provide faster turnaround times from requirements to implementation and offer a variety of capabilities that allow an applications developer to focus on the task rather than on the technical details. The advantages and disadvantages of each language (TPL and 4GL) were presented. After viewing the differences between the two languages, it was observed that, MIS soon would be implemented using both types of languages but the trend is heavily in the direction of 4GLs.

It was pointed out that the forces behind the move to 4GLs are managerial rather than technical. Integrating 4GLs into the traditional MIS department was discussed and it was observed that the largest impact this will have is in the software development methodology used. Due to their flexibility and ease of use, 4GLs enable developers to use prototyping for requirements specification as opposed to rigid design methodologies. The future direction of 4GLs was also indicated.

Overall, our findings are that 4GLs are here to stay until 5th generation languages evolve. For this decade 4GLs are a valid and useful concept in the software development arena.

REFERENCES

[1] Martin, James. Fourth-generation Languages. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985, p. xi.

[2] Cronin, Dan and Kathleen Bennett. SQL*Forms Designer's Tutorial Version 2.3. Belmont, CA: ORACLE Corporation, 1987, p. 3-11.

[3] Tillmann, George. "Prototyping For The Right Results." Datamation (April 1, 1989), p. 42.

[4] Luqi. "Software Evolution Through Rapid Prototyping." Computer (May 1989), p. 13.

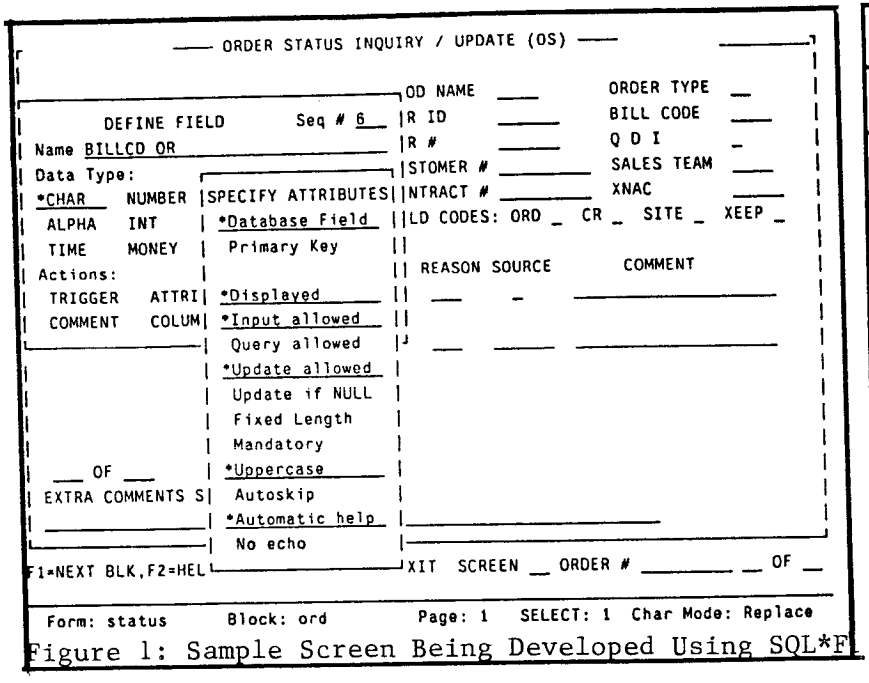


Figure 1: Sample Screen Being Developed Using SQL*F

Advantages	Disadvantages
Firm requirements aren't needed before coding starts.	Run slow - cannot be used in a real-time environment
Users can quickly see what their systems functionality will be and refine requirements based on this	May not be flexible enough to perform tasks that are requested of them.
Developers focus on results rather than on details of language - developers can be more business process oriented rather than programming experts	Visual thinking leads to coding / designing at the same time. Can lead to poorly designed systems which need to modified later. Significant for production systems only
Specify *what* to do rather than *how* to do it	
Can be used for visual thinking	
Applications take days / weeks to develop and maintain	
Maintenance is easier, less time consuming	
Numerous built in capabilities - scrolling regions, aggregates, etc	

Table 1 Advantages and disadvantages of fourth generation languages

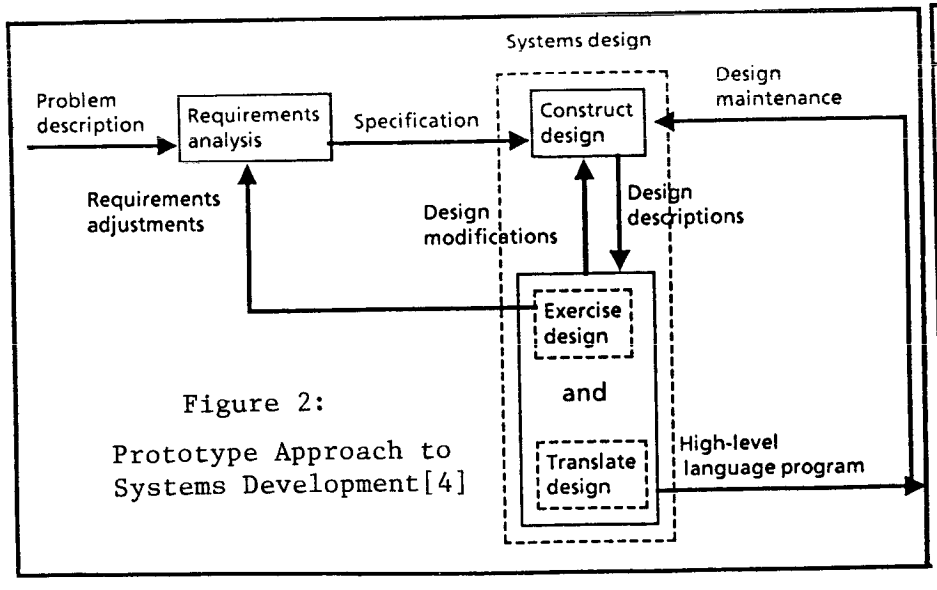


Figure 2:

Prototype Approach to Systems Development[4]

Advantages	Disadvantages
Run faster - can be used in many production and realtime environments	Need solid specifications before beginning to code since changes are difficult to incorporate
More flexible to handle special requirements	Applications typically take months / years to develop
	Cannot be used for visual thinking
	Cannot be used for prototyping
	Difficult and time consuming maintenance
	Applications developers must be programmers
	Users tell language *how* to do something

Table 2- Advantages & disadvantages of traditional programming languages

INCREASING THE VALUE OF END-USER DEVELOPED APPLICATIONS: AN INFORMATION CENTER CONCERN

J. K. Pierson
Karen A. Forcht
Faye P. Teer
James Madison University

ABSTRACT

If an application developed by an end user can be reused by others in the same organization, the value of the application to the organization is increased. One key to the increased use of existing applications, and thus increased value to an organization of the applications, is documentation. Deciding upon the appropriate amount and selecting appropriate forms of documentation for end-user developed applications is not a simple task. It is a task that requires training. This paper presents a training plan for teaching end users when and how to document their applications. The plan is based on the results of a study of documentation of user-developed applications.

INTRODUCTION

As a computer on each office worker's desk becomes the norm rather than the exception, the number of applications developed by end users increases. A recent article in Computerworld [15] points out there is probably no problem that is solvable by a computer that has not been reduced to code by someone, someplace. A substantial amount of that code has been written by end users. The value of computer application programs developed by end users can be increased dramatically if they can be used by other personnel. One key to increased use of existing applications, and thus increased value to an organization, is documentation. For instance, an undocumented spreadsheet developed at considerable expense to an organization by an end user can be used only by the developer. If the developer documents the application, however, anyone within the organization with the same need can use it.

In addition to increasing the use of existing application programs, documentation is important as a quality assurance technique. Particularly for those applications that have a significant potential impact on the organization and/or those that must be revised often, appropriate documentation procedures are essential. Not all end-user developed computer applications need the same amount or the same forms of documentation. A spreadsheet, for example, that is initially developed to be used by only one or a few employees and has no potential financial impact on the organization does not need the same amount of documentation as one that will be used by many individuals as the basis for significant financial decisions. Additionally, the forms of documentation appropriate for one type of computer application are not necessarily appropriate for another; that is, a spreadsheet needs different forms of documentation than does a database application. Deciding upon the appropriate amount and selecting appropriate forms of documentation for end-user developed applications is not simple. It is a task for which end users require training.

TRAINING END USERS TO DOCUMENT APPLICATIONS

Information centers responsible for training computer end users exist in most organizations today. As part of their training, end users should learn how to document their applications with the aim of achieving maximum benefits for the organization from the development costs. Training involves getting end users to recognize documentation as a two-step process: evaluating the application for the amount of documentation required, and choosing the appropriate forms of documentation for the type of application.

A recent survey of user-computing managers in the New York City area [32, 33] provides a basis for developing training guidelines for documentation containing the two steps outlined below.

Step 1: Evaluate Application for Amount of Documentation Required

Factors affecting the amount of documentation required for an end-user developed application are listed in Table 1 in rank order of importance as obtained from the survey.

The effect of each of the factors is briefly examined below:

- -Maintenance requirements. The first-place ranking of this factor is no surprise. For applications that will require frequent or long term maintenance, adequate documentation is very important.
- -Use of output by another system. The second place ranking is also not surprising. Management concerns for the need for documentation of user-developed applications that produce output used by other systems have been noted before [37].
- -Financial impact. An application with the possibility of a great financial impact can be assumed to have higher documentation needs than one with little financial impact regardless of whether its scope is personal, departmental, or organizational [37].
- -Scope of application. Logically, the larger the scope of the application the higher the documentation requirements. Management concerns for departmental and organizational applications are greater than for personal systems, increasing the need for documentation [37].
- -Complexity of application. It is plausible to assume that the more complex an application, the greater its documentation requirements. Another fact that must be recognized is that the complexity of user-developed applications tend to increase over time [17].
- -Turnover of user personnel. Anecdotal evidence exists that many user-developed applications are abandoned when the user-developer and/or users leave an organization. Therefore, the higher the turnover rate, the higher the documentation requirements.
- -Security requirements. Security requirements of user-developed applications are a vital management concern and have a direct impact on documentation. Security policies should be formulated not only for the application themselves but also for the documentation so that breaches do not occur at any level [16].
- -Audit requirements. Systems that will be audited require not only adequate documentation but also adequate testing of documentation to assure that the necessary audits can be carried out in a timely manner.
- -Type of application. Responses indicate that the type of application, i.e., modeling, data analysis, simple query, report generation, transaction processing or graphics, affects the documentation requirements of the application.

-Number of locations of use. If an application is to be used in many locations, it seems reasonable to assume that documentation requirements are higher than if the application is to be used in only one or a few locations.
-Number of users. Related to the factor just discussed is number of users of an application. Again, it seems reasonable to assume that as the number of users increases so does documentation requirements.
-Life expectancy of application. A logical conclusion is that applications developed for continued use require more documentation that those with a limited expected life.
-Development tool used. Some software development tools are more self-documenting than are others. Encouraging or mandating the use of self-documenting tools has been suggested as a means of facilitating control of user-developed applications [35].
-Anticipated reuse of application. Many programs intended as "throw away code" are, in fact, used over and over. This suggests that, at the very least, minimal documentation should be encouraged for all applications, whether or not the developers anticipate reuse [8].
-Frequency of use of application. Because of the limitations of human memory, infrequently used applications require more documentation than those that are frequently used. Even developers sometimes lose the benefits of their applications because they forget the intricacies of the systems and are dubious of their use [26].

The factors are further analyzed in Figure 1 where they are again listed in rank order obtained from the New York survey. Short descriptions for each factor illustrate computer application characteristics at each end of an amount-of-documentation-required continuum from high to low. By using the descriptions given in Figure 1, the end user can evaluate the computer application being developed for its documentation needs. For instance, the end user will evaluate an application as having high documentation needs if it will require frequent maintenance, will be used by a number of users in positions where historically there is high turnover, and is organizational in scope. On the other hand, an application that is personal in scope and is intended for one-time use will be evaluated as having low documentation requirements.

Step 2: Select Appropriate Forms of Documentation

If the outcome of an evaluation of an application is that it does require extensive documentation, the next step is to choose the appropriate forms of documentation. Different forms of documentation are appropriate for different types of applications. Several categories or classifications of computer applications exist. In the study of New York City user computing managers, the categories of applications used were:

--modeling: financial and other analytical models parameterized to allow "what if?" questions. Often spreadsheet or financial modeling software are used for these applications.
--data analysis: applications that use statistical packages such as SAS or SPSSx.
--simple query: applications that extract data from a file or database with no calculations made using the extracted data.
--report preparation: applications that extract data from a file or database, and include calculations (such as totals) made on the extracted data.
--transaction analysis: applications that update files created and controlled by user.

--graphics: applications that require the use of graphics software and produce charts or graphics.

The user computing managers surveyed recommended different forms of documentation for different classifications of applications. For instance, the top-ranked documentation form for graphics applications was hardware/software requirements, a form not ranked in the top five for any other type of application.

The five (or six, in case of a tie) top-ranked forms of documentation for each of the six categories of end-user applications appear in Figure 2. The percentage of user computing managers recommending each form is shown in the figure.

A summary of the top-ranked documentation forms for different categories of end-user developed applications appears in Table 2. The forms of documentation ranked somewhere in the top five for all application categories are:

--User's instructions
--Input data source, description, and layout
--Output destination, description, and layout

The remaining forms are ranked in the top five of some, but not all of the application categories:

--Narrative description
--Security requirements
--Testing procedures and test data
--Hardware/software requirements
--Program code, procedures, and/or formula listings
--Internal documentation

CONCLUSION

Adequate documentation of applications written by end users is important for several reasons:

1. to enable others in the organization to use the applications,
2. to facilitate revisions of applications that require frequent maintenance, and
3. to serve as a quality assurance tool in the development and testing phases of the development process.

Personnel who develop their own computer applications should receive training in evaluating their applications for the amount of documentation required and in choosing the appropriate forms of documentation for specific types of applications. Five of the most important factors to be considered in evaluating the amount of documentation required are:

--Maintenance requirements
--Use of output by another system
--Financial impact
--Scope of application
--Complexity of application

For those applications developed by end users that require a substantial amount of documentation, the following forms are recommended:

--User's instructions
--Input data source, description, and layout
--Output destination, description and layout
--Other forms of documentation appropriate for the application type

Through proper training, end users can learn guidelines for applying standard documentation procedures to the applications they develop. Proper documentation will increase the utilization of existing application programs and thus increase their value to the organization.

REFERENCES

Provided by authors on request.

Importance	Amount of Documentation HIGH <----------> LOW	
GREATEST	Frequent maintenance	Infrequent maintenance
	Output used by another system	Output not used by another system
	Financial impact significant	Financial impact insignificant
	Scope of application: Departmental or Organizational	Scope of application: Personal
	Complex application	Simple application
	High user turnover	Low user turnover
	High security requirements	Low security requirements
	High audit requirements	Low audit requirements
	Type of application: Transaction processing, Data analysis, or Report preparation	Type of application: Simple query, Graphics, or Modeling
	Used in many locations	Used in few locations
	Many users	Few users
	Long life expectancy of application	Short life expectancy of application
	Development tool with no self-documentation feature used	Development tool with self-documentation feature used
	Anticipated reuse of application	No anticipated reuse of application
LEAST	Infrequent use	Frequent use

Figure 1. Factor Importance and Amount of Documentation

Type of Documentation	Type of Application Modeling	Data Analysis	Simple Query	Report Prepar.	Transaction Process.	Graphics
User's instructions	91	87	87	93	93	91
Narrative description	91	85	72	91	-	89
Input data source	85	78	74	85	93	83
Output destination	85	78	70	85	93	83
Security requirements	-	-	63	76	93	-
Testing procedures	-	76	-	-	91	-
Hardware/software	-	-	-	-	-	93
Program code, etc.	80	-	-	-	-	-
Internal documentation	-	76	-	76	-	-

Figure 2. Percent of managers recommending a form of documentation

Table 1. Ranking of Factors Affecting Amount of Documentation Required

Ranking	Factor
1	Maintenance requirements
2	Use of output by another system
3	Financial impact
4	Scope of application--personal, departmental, or organizational
5	Complexity of application
6	Turnover of user personnel
7	Security requirements
8	Audit requirements
9	Type of application--modeling, data analysis, simple query, report generation, transaction processing, or graphics
10	Number of locations of use
11	Number of users
12	Life expectancy of application
13	Development tool used
14	Anticipated reuse of application
15	Frequency of use of application

Table 2
Rankings of Documentation Forms Recommended
for Six Classifications of End-User Developed Applications

	Type of Application					
Type of Documentation	Modeling	Data Analysis	Simple Query	Report Prepar.	Transaction Process.	Graphics
User's instructions	1	1	1	1	1	2
Narrative description	1	2	3	2	-	3
Input data source	3	3	2	3	1	4
Output destination	3	4	4	3	1	4
Security requirements	-	-	5	5	1	-
Testing procedures and	-	5	-	-	5	-
Hardware/software	-	-	-	-	-	1
Program code, etc.	5	-	-	-	-	-
Internal documentation	-	5	-	5	-	-

IMPROVED MAINTENANCE TECHNIQUES: THE IMPACT OF SOFTWARE MAINTENANCE TOOLS ON THE APPLICATIONS PORTFOLIO

Emerson Maxson
Boise State University

ABSTRACT

The literature on software maintenance tools has emerged in the last six years. It suggests that substantial savings in maintenance costs can be derived from the application of these tools to a company's portfolio of application programs. Most of what has appeared on this subject is anecdotal or descriptive in nature and therefore generalizations from any of this work is not possible. The current study investigates the use of these tools across a variety or organizational settings. This approach will deliver information about patterns of acceptance or rejection of these tools by the information systems community. Even though the results of this study are small in number, they are beginning to tell an interesting story about the use of two classes of software maintenance tools.

INTRODUCTION

The maintenance of software consumes 2 out of every 3 dollars in Information Systems budgets in organizations, according to the experts in the field such as Boehm, Jones and others (2, 5, 18, 22, 26, 27). Maintenance is vitally important to the success of the Information Systems function within an organization, yet, there has been little research relating to software maintenance as has been described by Schneidewind and Hale (17, 33). According to Schneidewind no articles on maintenance appeared in IEEE Transactions on Software Engineering between August, 1985 and November, 1986 and only a few were published before that time. In a paper presented at the 1988 Software Maintenance Conference Hale found only 51 empirical articles from a broad range of journals whose topic was maintenance.

Software maintenance tools that may improve the productivity of the software maintenance task have been described in anecdotal fashion in the last 6 years (1, 3, 9, 10, 11, 12, 13, 14, 19, 21, 24, 26, 31). These articles have described a wide variety of tools, but the ones that have received the majority of attention are code restructuring and code analyzer tools. Code restructuring tools accept unstructured code as input and produce as output a structured program with the same functionality. A list of these tools follows: 1) RECODER, from Language Technology, 2) RETROFIT, from Peat Marwick, 3) SUPERSTRUCTURE, from Computer Data Systems, Inc., 4) ADPAC/SS, from ADPAC, 5) COBOL APPRENTICE, 6) COBOL RESTRUCTURING FACILITY, from IBM. Code analyzer tools accept programs as input and produce reports that help a programmer see and understand the logic that previous programmers put into the program. A list of these tools follows: 1) INSPECTOR, from Language Technology, 2) PATHVU, from Peat Marwick, 3) SCAN/COBOL, from Computer Data Systems Inc., 4) COBOL/METRICS, from Computer Data Systems Inc., 5) SCOREBOARD, from Travtech, 6) AUDITEC, from Maintec Inc., 7) VIA/INSIGHT, from Viasoft. Tools such as these are designed to help manage the software applications portfolio.

In an industry survey carried out in 1986, Brancheau and Wetherbe (7) identified managing the applications portfolio as one of the top ten technology/application issues for Information Systems practitioners.

RESEARCH QUESTIONS

The present study investigates the circumstances under which code restructuring and code analyzer tools have been utilized across a variety of organizational settings. These tools have been sold to a large number of organizations over the last few years and the experience of these organizations needs to be measured and understood to guide future research as well as future use of these tools by practitioners. Questions this study has investigated include: 1) size of applications portfolio, 2) condition of the portfolio as measured by age and structuredness of the programs, 3) experience in using restructuring and analyzer tools, and 4) measurement of the current applications backlog.

RESEARCH METHOD

A questionnaire was chosen to gather information about the primary research questions. The survey was mailed to the 100 firms selected by Computerworld in 1989 as those most effectively investing in information systems (THE COMPUTERWORLD 100). This sample was chosen because these firms have shown leadership in the effective use of information systems and should therefore represent the leading edge in the development and use of new software tools. If there is going to be significant use of this type of tool, a pattern should appear first in this sample.

CURRENT RESULTS

As of the publication deadline 20% of the sample had completed and returned the questionnaire. This response rate might suggest that many recipients were unfamiliar with the technology. Or that the topic of software maintenance still is not as interesting as new development and such tools as CASE. However, the results that will be reported must be considered in the light of a 20% response rate.

A broad spectrum of industries are represented in the results, see below for details.

Industry Classification	Percentage Responding
Financial Institutions	25%
Mining/Oil	15%
Manufacturing	20%
Aerospace	10%
Transportation	10%
Utilities	10%
Chemicals	5%
No Response	5%

COBOL was reported as being in current use by 85% of the firms and the reported applications portfolio size ranged from a high of 75 million lines of code to a low of 1 million with an average of 12 million. The average was skewed to the high side because of the one large portfolio.

Code analyzers are being used by 45% of the respondents and code restructurers are being used by 40%. The major impediment to the use of code analyzers as reported by 45% of the respondents was their lack of distinguishable benefits. The major impediment to the acceptance of code restructurers as reported by 40% of the respondents was that the code style generated by the restructuring tool did not fit company style standards.

The applications portfolio as measured by age displays a rather flat distribution with 19% of the programs less than three years old, 26% three to six years old, 25% six to ten years old and 30% over ten years old. The portfolio as measured by structure shows

a markedly different distribution with 9% of the programs being very unstructured, 18% unstructured, 34% modular and 39% structured. These numbers are interesting in that 30% of the portfolio is over ten years old but only 9% is very unstructured. It might have been expected that the age and structuredness of the portfolio would more closely correlate.

The applications backlog data show that 12% of the organizations have a backlog of less than six months, 35% have a backlog of six months to one year, 24% have a backlog of one to two years and 29% have a backlog greater than two years. This pattern suggests that the applications backlog is a problem that exists even in those companies who have been very effective in investing in information systems.

Sentiment about replacing COBOL with 4th or 5th generation languages is widely split with 58% agreeing or strongly agreeing and 42% disagreeing or strongly disagreeing and no respondents having no opinion. This split suggests there are very strong feelings on both sides of the question and this issue will remain with us for some time and as Brooks states so well in his April 87 paper, "we see no silver bullet" (8).

Maintenance, according to our respondents, is growing faster than new applications, with 71% strongly agreeing or agreeing and only 29% disagreeing or having no opinion. This confirms that maintenance as an ongoing issue for management attention is well deserved.

The biggest surprise in the results was that the structured design and programming methodology use is rather evenly spread between Yourdan at 20%, Martin at 15%, Warnier/Orr at 10%, Jackson 5%, other 10% and no response 40%. It had been expected that the Yourdan methodology would be much more widely used than all the rest and that the no response category would be very small.

DISCUSSION

These results begin to suggest that software maintenance tools are being used even though this technology is still very young and the current applications portfolios contain only 9% very unstructured code. Also these tools are spread among a broad spectrum of industries with the financial institutions being the most active. If this emerging pattern spreads beyond the COMPUTERWORLD 100, maintenance tools such as these should become more important as part of the ongoing software renewal/development mix.

The academic community needs to respond to this emerging pattern by developing case studies of the various ways companies have successfully managed the use of these tools. These studies could track the payoff of the application of software maintenance tools over time and determine their impact on such things as the applications backlog and the cost of maintenance.

REFERENCES

1 Abi, Ray, "Software Maintenance: Tools and Techniques", System Development, August, 1988, Pages, 3-6.

2 Alper, Alan, "COBOL Monitor eases code maintenance", Computerworld, March 7, 1988, Page, 30.

3 Babcock, Charles, "Restructuring eases maintenance", Computerworld, November 30, 1987, Page, 19, 22.

4 Babcock, Charles, "COBOL restructuring debated", Computerworld, January 19, 1987, Pages, 25, 27.

5 Boehm, Barry, "Developers must design in Maintainability", IEEE Software, May, 1988, Pages, 104-105.

6 Boehm, C. and Jacopini, G., "Flow diagrams, Turing machines, and

languages with only two formation rules", Communications of the ACM, May, 1966, Pages, 366-371.

7 Brancheau, J and Wetherbe, "Key issues in Information Systems", MIS Quarterly, January, 1987, Pages, 23-45.

8 Brooks, Frederick, "Essence and Accidents of Software Engineering", IEEE Computer, April, 1987, Pages, 10-19.

9 Bush, E., "The Automatic Restructuring of COBOL", Conference on Software Maintenance, November, 1985, Pages, 35-41.

10 Canning, R., "Rejuvenate your old systems", EDP Analyzer, March, 1984, Pages, 1-16.

11 Canning, R., "Tools to rejuvenate your old systems", EDP Analyzer, April, 1984, Pages, 1-16.

12 Carlyle, Emmett, "Can AI save COBOL?", Datamation, September 15, 1985, Pages, 42-43.

13 Gallant, John, "IBM spotlights restructuring", Computerworld, December 9, 1985, Pages, 19, 25.

14 Gamble, Sharon, "What a tangled web, we weave", ICP Software Review, November, 1986, Pages, 28-32.

15 Guillo, Karen, "Steady as she goes", Datamation, January 15, 1987, Pages, 37-40.

16 Guillo, Karen, "Automating COBOL Support", Datamation, January 1, 1988, Pages, 19, 22.

17 Hale, D and Haworth, D., "Software Maintenance: A Profile of Past Empirical Research", Proc. Conf. on Software Maintenance, 1988, IEEE Computer Society Press.

18 Harrison, Richard, "Maintenance Giant Sleeps", Computerworld, March 9, 1987, Pages, 21, 81.

19 Horton, L., "Tools are an alternative to 'playing computer'", Software Magazine, January, 1988, Pages, 58-67.

20 Hummer, Thomas M.,"Maybe 4GLs Can Kill COBOL", Information Center Magazine, February, 1990, Page 40.

21 Jander, Mary, "COBOL'S Salvation?", Computer Decisions, January, 1989, Pages, 67-71.

22 Jones, C., Programming Productivity, McGraw-Hill, 1986.

23 Kolodziej, Stan, "COBOL shapes up", Computerworld (FOCUS), January 7, 1987, Pages, 13, 14.

24 Kull, David, "The Aging Systems Saga", Computer Decisions, January, 1989, Pages, 42-47.

25 Lientz, B. P. and Swanson, E. B., "Problems in application software Maintenance", Communications of the ACM, December, 1981, Pages, 763-769.

26 McShane, Patricia, "Born again systems, Part II", Computer Decisions, April 8, 1986, Pages, 80-86.

27 Parikh, Girish, Programmer Productivity, Reston Publishing Co., 1984, Pages, 20-23, Reston, Virginia.

28 Parikh, Girish, "Restructuring your COBOL programs", Computerworld (FOCUS), February 19, 1986, Pages, 39-42.

29 Parikh, Girish, "Assault on tangled code", Computerworld, June 29, 1987, Page, 54.

30 Parikh, Girish, "COBOL restructuring engines clean up Spaghetti code", Computerworld, March 31, 1988, Pages 59-64.

31 Phillips, Roger, "Rating the Maintenance Tools", Computerworld Extra, June 20, 1988, Page, 43.

32 Pizzarello, Antonio, Development and Maintenance of Large Software Systems, Wadsworth, Inc., 1984, Pages, 27-29, Belmont, California.

33 Schneidewind, Norman, "The State of Software Maintenance", IEEE Transactions on Software Engineering, March, 1987, Pages, 303-310.

34 Sneed, H.M.and Jandrasics, G., "Software Recycling", Conference on Software Maintenance, 1987, Pages, 82-90.

35 Sokol, Mark, "Why 4GLs Cannot Kill COBOL", Information Center Magazine, December, 1989, Page 40.

36 Snyders, Jan, "Can COBOL live forever?", Infosystems, August, 1986, Page, 54.

37 Weber, Ron, et. al., "Research on Structured Programming: An Empiricists Evaluation", IEEE Transactions on Software Engineering, July, 1984, Pages, 397-407.

38 Zvegintzov, Nicholas, "Immortal Software", Datamation, June 15, 1984, Pages, 170-180.

39 Zvegintzov, Nicholas, "High Noon Part II: The Quest for a true test of software maintenance tools", Software Maintenance News, August, 1986, Pages, 1-2.

CRITICAL EXPERTISE MATRIX: STRATEGIC TRAINING PLAN FOR INFORMATION SYSTEMS DEPARTMENTS

Mark W. Smith
Department of Computer Technology
Purdue University

ABSTRACT

Training faculty for the 1990's will be a difficult task. An "education/expertise" gap exists in many Information Systems (IS) departments already and will grow even wider as new hardware, software, and methodologies arise. To help IS departments plan for their current and future training needs, a strategic planning guide called the Critical Expertise Matrix (CEM) was developed. CEM is simple to use and can be modified for different departments and colleges. It can help to quantify training needs often considered to be a very subjective process. Using CEM can also help the IS department set training priorities. A sample department is presented along with a complete description on how to apply CEM in an IS department setting.

INTRODUCTION

How does your department determine what its "education/expertise" gap will be for the 1990's and beyond? As Information Systems (IS) Departments in many colleges and universities continue to grow, both in the number of students and the number of faculty, and as their curriculums continue to change, IS Departments must develop a well thought out professional development/training program for their faculty in order to meet these new demands. New equipment, courses, techniques and methodologies, along with faculty vacancies will require extreme flexibility as well as critical skills from the faculty in an IS Department.

Unfortunately, very little planning is done by many IS Departments to solve the "education/expertise" gap. Funds are limited and must be applied judiciously. Thus, a strategic plan is needed to facilitate the growth, skill acquisition, and experience necessary to carry an IS Department through the 1990's. To meet the training and expertise levels needed, a simple process called the Critical Expertise Matrix (CEM) was developed to guide an IS Department in its assessment of training needs and priorities.

BACKGROUND

Although training is something most people take for granted, it is the major purpose of a college or university. One of the goals of colleges and universities is to help students develop marketable job skills through diversified and comprehensive technical and college transfer courses. At the IS Department level this translates into providing majors with the computer skills they will need to gain employment as computer professionals.

In many IS Departmental Training Plans (when they exist!) an overall training goal is often stated similar to the following:

> to have qualified instructors as required to meet the educational instruction level necessary to provide quality training in all courses in the IS curriculum for the student.

To stay current the faculty must continually learn new skills. To

provide technical training to students at the level required by business, training must be "state of the art." Thus the goals and objectives are most often met by training received in new and developing areas of computer technology.

Recently, this technology has taken a quantum leap forward at most colleges and universities. Not only are there new requirements to teach courses using mainframes and minicomputers, but with the prolific number of microcomputers and rapidly expanding networks, the need for obtaining new skills currently unavailable in many faculty ranges of experience and expertise becomes critical. Table 1 illustrates some of the areas a "sample" Information Systems department must provide training and expertise. Any table item listed with an asterisk (*) next to it indicates that the skill is not currently held by IS faculty or only one faculty member has the skill, as such it is a critical skill.

Table 1.

REQUIRED SKILLS FOR INFO. SYSTEMS FACULTY (SAMPLE DEPARTMENT)

	Computer Hardware
	IBM 370
*	IBM AS/400
*	IBM 9370
*	IBM RT (RISC)
*	IBM PS2 MODEL 80
	IBM PS2 MODEL 30
	IBM PC/XT
	DEC VAX CLUSTER
*	DEC MICROVAX II
	DEC RAINBOW MICROS
	ETHERNET NETWORK

As noted in the table, there are many skills either not held currently or by only one member of the faculty. These skills could be labelled as critical in that they must be met by either hiring new faculty members with these skills or by training current faculty. Even though new faculty might be hired with some of the skills, it is highly unlikely that they could possess all the skills, let alone be able to teach the many courses requiring those skills. Therefore, using concepts found in most strategic management courses, a method was developed called the Critical Expertise Matrix which can help identify weaknesses and strengths as well as help in prioritizing training needs.

Using the Critical Expertise Matrix (CEM) will not only help identify training areas where deficiencies exist, but it can also be used as a planning and tracking document both for the department and the faculty member.

CEM METHODOLOGY

Analyzing training needs for faculty is a difficult and subjective process. To improve upon this process an evaluation of where the IS Department (as a whole) is currently, and where they want to be in 3-5 years, should be undertaken. A major step is to identify strengths and weaknesses, as a unit, and determine the priority of training necessary to overcome them. This can be done using the Critical Expertise Matrix (CEM).

The matrix, with the Critical Knowledge Areas (CKA) for the "sample" IS Department is identified, along with the instructions and appears in figure 1. They are a consolidated list from the department faculty, and a Computer Technology Advisory Committee which was made up of approximately 10 data processing managers from a cross-section of businesses. Also included was information gained from private discussions with other data processing personnel in the local area. The Advisory Committee was responsible, for instance, in the inclusion of a course in CICS being added and its addition as

a Critical Knowledge Area in the CEM. The CEM method helps to organize areas of needed expertise and cross them with individual levels of expertise. It then helps to further identify critical knowledge areas (CKA's) and which critical knowledge areas need immediate attention. To do this the matrix is simply filled in.

The first step in CEM is to fill in the CRITICAL KNOWLEDGE AREAS (CKA's). Put one item per line as shown in figure 1. Next, assign a weight to each item as to its relative importance to the mission of the curriculum. This could include such things as how many courses and sections in this area are to be taught and how hard it would be to find a part-time faculty member if none of the full-time faculty have the skill. A "0" would mean that the item is nice to have but not necessary and essentially would have no impact on the mission/curriculum. On the other hand, a "3" would be a critical knowledge to have for the successful teaching of the curriculum. In other words, without it a course or several courses in the curriculum could not be taught. (Use the guidelines listed at the bottom of figure 1. - WEIGHT FACTOR FOR CKA)

Next, list all the instructors and their Knowledge level for each of the CKA's. In the figures and appendices they are listed as A, B, C, D, E, and F. A knowledge level of "0" would obviously indicate no knowledge, a "1" would mean that the instructor has very limited knowledge in the area gained either by experience or by formal training but not both. A "2" would mean the teacher is knowledgeable enough to teach the course from experience or from formal training but again does not possess both. A "3" means the instructor has both formal training and experience and is in essence an expert in the critical knowledge area. Obviously, this scale could be changed by the individual IS Department or college to better conform to their unique circumstances. (Again, see the bottom of figure 1. for guidelines - INSTRUCTOR KNOWLEDGE LEVEL)

Once the instructors are ranked in each area the total knowledge level for that area is obtained by totaling each instructor's knowledge level together. Place this in the total knowledge column. Divide this number by the total number of instructors which will give the actual knowledge level for the department for that particular CKA. Put this value in the actual knowledge column. Compare this number with the weighted number for the CKA. As a rule of thumb, if the actual knowledge level is less than half the weighted value in the Weight Factor column, it may appear that this is a problem area and warrant immediate attention for training purposes. Figure 2 shows an abbreviated example of how the matrix and its evaluation would work based on the "sample" IS Department. Remember that we are comparing the actual knowledge to the weight factor. CKA 2 and 3 are probably not a high priority for training while CKA 5 is. CKA 1 and 4 are important and should be considered for training as soon as training for CKA 5 has been met.

A variation of the CEM form in figure 2. would be the addition of another column called the priority factor column at the far right of the form. This column would contain a percentage number which would be found by dividing the actual knowledge column by the weight factor column. The next step would be to arrange the CKA's in order by this priority factor giving you some idea of priority of training needs (yes, CEM lends itself to an electronic spreadsheet!)

Of course the CEM tool is only meant to help quantify needs for training. It doesn't take into account part-time instructors and is obviously subjective with respect to the weighting of the CKA and the instructor's knowledge level. However, it is felt that by using this tool some level of priority and needs requirements can be determined to help

put together a meaningful training plan.

OTHER FACTORS

It should be remembered that any plan must be fluid and adaptable to changing climates. The CEM could easily evolve into a college-wide system. The development of the IS Departmental training goals through the use of the CEM will be the most difficult phase and will require a lot of refinement as time passes. It might even be advisable to survey as many businesses as possible to find out what they think their long range (3-5 years) plans will be.

CONCLUSION

The formalizing of training and professional growth will enhance the individual faculty member's ability to maintain currency in his/her field as well as plan and build for expansion in the future. The Critical Expertise Matrix can help an IS Department determine their training needs. In the computer field change is inevitable. Planning and training for that change is critical for an educational institution's survival and growth!

CRITICAL EXPERTISE MATRIX
SAMPLE IS DEPARTMENT

CRITICAL KNOWLEDGE AREA (CKA)	WGT FACTOR	KNOWLEDGE LEVEL OF INSTRUCTORS A	B	C	D	E	F	TOT KNOW.	ACTUAL KNOW.
HARDWARE KNOWLEDGE									
1. IBM 370									
2. IBM AS/400									
3. IBM 9370									
4. IBM RT									
5. IBM PS2 MODEL 80'S									
6. IBM PS2 MODEL 30'S									
7. IBM PC/XT/AT									
8. DEC VAX CLUSTER									
9. DEC MICROVAX II									
10.DEC RAINBOW MICRO'S									
11.ETHERNET NETWORK									

INSTRUCTOR KNOWLEDGE LEVEL	WEIGHT FACTOR FOR CKA
0=No Knowledge	0=CKA not important
1=Some Knowledge (some formal training or some experience)	1=CKA plays minor roll in meeting teaching responsibilities
2=Knowledgeable (formal training or job experience)	2=CKA is important, some course or curriculum objectives may not be met
3=Very Knowledgeable (formal training and job experience)	3=CKA is critical, required to meet curriculum objectives

Figure 1.

CRITICAL EXPERTISE MATRIX

CRITICAL KNOWLEDGE AREA (CKA)	WGT FACTOR	KNOWLEDGE LEVEL OF INSTRUCTORS A	B	C	D	E	F	TOT KNOW.	ACTUAL KNOW.
HARDWARE KNOWLEDGE									
1. IBM 370	3	2	3	1	1	1	2	10	1.7
2. IBM PS2 model 30's	3	3	3	2	3	3	3	17	2.8
3. IBM 9370	2	1	1	2	3	1	2	10	1.7
4. IBM RT	1	2	2	0	0	0	0	4	.67
5. IBM PS2 MODEL 80'S	3	2	3	0	0	0	0	5	.83

Figure 2.

THE IMPLICATION OF ORGANIZATIONAL STRUCTURE ON THE USAGE OF THIRD/FOURTH GENERATION LANGUAGES

Karen J. Ketler
Eastern Illinois University

Susan M. Moncada
Indiana State University

This paper compares and contrasts third and fourth generation languages in areas such as language structure, skill requirements, work-place environment and user/customer importance. The critical variables in the selection of a third or fourth generation language are also variables in the selection of the functional or product organizational structure. The correlation between organizational structure and the usage of a computer generation language is explored.

INTRODUCTION

The goal of any language is to communicate. In the case of computer languages, the communication is between people and computers. Traditionally, people have been taught to communicate with computers using the language of the computers [3]. The problem associated with this communication with computers has been identified as the "linguistic distance between the effective language of the computer and the language used by human beings to describe the problem" [9]. Computer language has evolved over time in order to reduce linguistic distance.

THIRD GENERATION LANGUAGES VS. FOURTH GENERATION LANGUAGES

The 1960's brought forth the third generation of languages (3GLs). These languages, known as high level languages, describe processing procedures. The instructions in these languages resemble the language of the person. Although these languages generally make programming a problem easier and faster than the assembly languages, they still require a large number of lines of detailed code for a program [10]. They are designed for use by a trained data processing professional. They are generally used in a structured environment on well-defined problems and require sequential, detailed logic.

The fourth generation of languages (4GLs) are the non-procedural languages. Designed for nonprogrammers, these languages "resemble the language of the subject matter in which the user has been trained" [9]. These languages describe the desired result rather than the details or the actions needed to achieve that result. They can provide an interactive dialogue as a guide for application development. They have powerful non-procedural verbs with English-like syntax. They can interact with numerous 3GLs. They are simple to learn and have query language capabilities for direct access to databases [5]. While the 3GLs are used for structured problems, the 4GLs are typically used for spontaneous, ad hoc reporting.

The 3GLs and the 4GLs are quite different. Some of the variables and situations on which these languages differ are itemized in Figure 1. These variables have been identified as significant characteristics affecting organizational and departmental structure [4]. Thus, there is a

correlation between the organization structure characteristics and the role of 3GLs/4GLs.

Various authors [1,3,8] have predicted the demise of the 3GLs while others [7] have forecasted that the 4GLs are just a passing fad. The role of the 3GL in information systems depends on a wide range of factors from the organizational and departmental structure to the skills and characteristics of the employees in information systems. The remainder of this paper will address the implications of organizational and departmental structure on the role of these languages.

ORGANIZATIONAL STRUCTURE

Organizational structure consists of two dimensions: (1) the structural dimension includes the internal characteristics of the organization and (2) the contextual dimension includes the characteristics of the entire organization. Important variables influencing these dimensions are detailed in Figure 2 [4]. These variables take on different values in the two basic organization forms: the function and the product. The functional structure groups people and activities by resources such as marketing or data processing. This structure is used in a stable environment with low uncertainty, because it is slow to respond to environmental changes. The strengths of the functional structure is internal efficiency with technical specialization and in-depth skill development. Policies and procedures are well-developed throughout the organization.

The product structure is a grouping based on organizational outputs. For each product output, all necessary resources, such as manufacturing, research and development, marketing, accounting and data processing, are grouped within a department. While this structure is suited to fast changing environments with emphasis on client satisfaction, it does sacrifice the in-depth technical skill development. Some duplication of efforts by each product unit exists. In addition, there is little standardization of policies and procedures across product lines.

Thus, the variables which affect the selection of an organizational structure (and indirectly in the selection of language generation) are detailed in Figure 3. In an effort to combine the advantages of functional and product organizational forms, the matrix structure was developed. In such a structure, each unit, such as data processing, reports to both product and functional managers. The matrix structure is generally used in an uncertain environment. It allows for the development of skills of the employees. The concept of dual authority is the major weakness of this structure.

The executive personnel who elect to organize the company in the functional structure are apt to emphasize the 3GL in their information systems. There is a strong correlation between the emphasis of the 3GLs and the functional organizational form. Skill development is emphasized. Processing and problems are routine. The atmosphere is generally structured with adequate documentation of policies and procedures.

Likewise, there is similarity between the customer emphasis of the product structure and the user emphasis in the 4GLs. Both react fast to spontaneous ideas and are capable of adapting to the non-routine requests.

The organizational structure influences the design of the information systems department which, in turn, may affect the role of the 3GLs. The three most common structures of data processing are centralized, decentralized and distributed.

DEPARTMENTAL STRUCTURE

With the development of the 3GLs, the concept of centralized data processing evolved in the 1960's. Hardware costs decreased as technology advanced. Total resources invested in computer system began to grow. Management began to look at the high cost of duplicate hardware, software development, staffing, and overhead costs. It was more efficient to centralize [2]. This thinking produced the separate data processing department.

At first, the functional form of organization appeared to make the most efficient use of the computer. With the arrival of standardized languages, the data processing department tackled the problems of incompatible procedures within the organization. With the 3GLs, the productivity of the technicians improved; however, the requests for information processing increased even more rapidly than productivity increased.

As the backlog of programming requests grew, managers started seeking alternative ways to meet their needs. In the early 1970's, decentralization began. The arrival of the new technology of mini- and micro-computers became a catalyst for decentralization. This development enabled each department within the organization to have its own information processing capability. In a totally decentralized system, all processing is local, and there is no data communications directly among systems. As a result, there is duplication of effort in the administrative functions. Decentralization created smaller data processing departments and less specialization within the department.

By the 1980's, with the enhanced telecommunication technologies and user-oriented languages, distributed data processing (DDP) evolved. Distributed data processing is characterized by physically separated computers which are interconnected through communications facilities [5]. Wolff and Litecky [11] defined DDP as the distribution of control and responsibility for computer resources to the levels within the organization where they are needed by the end-user.

A survey of large data processing facilities revealed that fifty-one percent were centralized, twenty-nine percent were decentralized, and twenty percent were distributed [2]. Along with the increased interest in DDP, there was an increase in the use of programming teams and the matrix organizational structure. Programming teams and the matrix structure allows the manager flexibility in the organization of work and people in order to create a positive atmosphere [6].

The literature suggests that there is a strong correlation between the centralized data processing department, the functional organizational structure and the importance of 3GLs. At the opposite extreme is the decentralized data processing department, the product organization structure and the importance of 4GLs. And, in the middle, is DDP and the matrix organizational structure. In this middle road lies the successful combination of the 3GLs and 4GLs.

BIBLIOGRAPHY

1. Abbey, Scott G. "COBOL Dumped." Datamation. January, 1984, pp. 108-114.

2. Blank, Mark M. "Microcomputing Comes of Age." Journal of Systems Management. June, 1986, pp. 32-37.

3. Cobb, Richard H. "In Praise of 4GLs." Datamation. July 15, 1985, pp. 90-96.

4. Daft, Richard L. Organization Theory and Design. St. Paul: West Publishing Company, 1986.

5. Davis, Gordon B. and Margrethe H. Olson. Management Information Systems. New York: McGraw-Hill Book Company, 1985.

6. Friedman, Andrew and Joan Greenbaum. "Wanted: Renaissance People." Datamation. September 1, 1984, pp. 134-144.

7. Grant, F. J. "The Downside of 4GLS." Datamation. July 15, 1985, pp. 99-104.

8. Harding, Elizabeth. "4GL Vies for COBOL's Heavyweight Crown." Data Management. November, 1985. pp. 33, 46.

9. Klerer, Melvin. User-Oriented Computer Languages: Analysis and Design. New York: Macmillan Publishing Company, 1987.

10. Martin, James. Fourth Generation Languages. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1985.

11. Wolff, Robert L. and Charles R. Litecky. "Distributed Data Processing Controls." The EDP Auditor. Fall, 1982, pp. 1-14.

FIGURE 1

Environment:	Routine 3GL	Nonroutine 4GL
Procedurally/Sequentially oriented:	Low 4GL	High 3GL
Documentation and Policies:	Low 4GL	High 3GL
Specialization of Skills/ Complexity:	Low 4GL	High 3GL
Structured Problems:	Low 4GL	High 3GL
Standardization of Language:	Low 4GL	High 3GL
Amount of training/education:	Low 4GL	High 3GL
Role of the User:	Low 3GL	High 4GL

FIGURE 2

(A)	Structural Dimension	-	an organization's internal characteristics.
	(1) Formulation	-	amount of written documentation, policies and procedures in an organization.
	(2) Specialization	-	extent organizational tasks are subdivided into separate jobs.
	(3) Standardization	-	extent similar work activities are performed in a uniform manner.
	(4) Complexity	-	number of activities/subsystems.
	(5) Centralization		hierarchical level having decision making authority. The higher the level, the more centralized the organization.
	(6) Professionalism		extent of employees' formal education and training.
(B)	Contextual dimension	-	characterize the whole organization.
	(1) Size		organization's magnitude is reflected in the number of people in the organization.
	(2) Organizational Technology	-	nature of the tasks in the production subsystem. Includes action, knowledge and techniques used to change inputs into outputs.
	(3) Environment		includes all elements (government, customers, supplies and the financial community) outside the boundary of the organization.
	(4) Goals		define scope of operations and desired relationships with employees and clients.

* Adapted from [4]

FIGURE 3

	Functional	Product
Environment:	Stable (3GL)	Unstable (4GL)
Procedurally oriented:	Product Low (4GL)	Functional High (3GL)
Documentation and Policies:	Product Low (4GL)	Functional High (3GL)
Specialization of Skills/ Complexity:	Product Low (4GL)	Functional High (3GL)
Structured Problems/ Technology:	Product Nonroutine (4GL)	Functional Routine (3GL)
Standardization:	Product Low (4GL)	Functional High (3GL)
Skill development: (Training/Education)	Product Low (4GL)	Functional High (3GL)
Role of the User/Customer:	Functional Low (3GL)	Product High (4GL)

CHANGING THE QUALITY ASSURANCE OUTLOOK OF SMALL AND MEDIUM SIZED BUSINESSES THROUGH THE USE OF COST-EFFECTIVE COMPUTING

Lawrence P. Huggins
J. Anthony Flynn
Manhattan College

ABSTRACT

With increased growth of international trade, customer quality expectations are increasing in the face of enlarged purchasing choices. For many small and medium sized businesses, often lacking sophisticated quality assurance know-how, this seems to create an insuperable barrier to competing on a level playing field of quality competence. In this age of relatively low cost personal computers, this is a strategic error. These companies can, through awareness and training, develop cost-effective quality assurance programs which are a structured blend of well-tested quality techniques combined with the application of personal computers to the management of this important competitive element. This paper addresses itself to key considerations necessary for small and medium sized businesses to provide themselves with competively priced computing capabilities that will permit them to function as fully equal competitors for consumers concerned with the relative quality merits of product choice. It depicts the sources of quality costs and demonstrates, from pre-production analysis through full production start-up, how small and medium sized businesses can use personal computers effectively to reduce the apparent advantages of larger competitors, foreign or domestic.

QUALITY IS FUNDAMENTAL TO COMPETITION

In today's global market place, product quality, whether actual or perceived, has become a competitive reality in every industry. From the production of consumables such as beer and chocolates to the design and manufacture of communications satellites, competition has taken on a worldwide dimension and customers are scrutinizing products with more sophistication and with an unforgiving eye for imperfections, and shoddiness in general. With the continued growth of international trade, customer quality selectivity can be expected to grow in the face of enlarged procurement choices. Indeed competition is becoming more of a creature of quality and service rather than of price as competition for raw materials grows world wide and wages in other countries begin to move towards parity. The stereo sound fidelity system that commanded a high price in the 1970s is no longer acceptable by today's standards. Automobile warranties of the 1940s, 50s and 60s of 1000 miles or 90 days performance would bring a look of disgusted disbelief to the faces of today's consumers. With the high level of imports, consumers have become progressively more demanding in their expectations. Even in the production of military hardware, on which considerations of quality have always been paramount, public awareness of the costs of failure have been translated into demands for congressional actions.

Quality assurance for all companies: Necessarily then, it appears fundamental to business survival that every manufacturer in the United States assure

itself that it has adopted a quality assurance posture consistent with the competitive realities of the marketplace. Product quality is not without its costs. For some these may be perceived as too high thus relegating them to a position on the low end of the quality spectrum. For many, especially small and medium sized businesses, which lack sophisticated quality assurance know-how, the cost of solid quality programs similar to those of large corporations may seem an insuperable barrier. Unfortunately then, many withdraw to compete only on the grounds of price and delivery. In this age of relatively low cost personal computers, this is an error with long-term implications.

Quality program needs: New awareness by most small and medium businesses for the development and implementation of quality programs are needed. The reality of today's technology is that all of the traditional objectives of quality control can now be available to small and medium size companies for relatively low costs. The objective of every business should be to manufacture products which obtain favorable customer acceptance and profitable results. A structured blend of well-tested quality techniques combined with the use of personal computers can give any business a quality system which will more than pay for itself.

KEY CONSIDERATIONS

Years of hands-on experience and distillation of literature on the subject suggests that there are some seven key areas in which work needs to be done to obtain the kind of product quality needed to compete in chosen market arenas. These are: (1) pre-production analysis, (2) product/process planning, (3) purchased materials control, (4) factory product/process control, (5) information feedback, (6) post production quality requirements, (7) quality training.

For a serious approach to establishing a quality assurance program, with or without the use of quality circles, it is important that all of these subjects be given a high level of management priority. In today's technologically "live" environment each of these areas can be aided by the use of existing state-of-the-art personal computer systems. Quality costs, composed as they are of the costs associated with prevention of defects, expenses attendant upon inspection and test, and the often monumental costs created by product failure in the factory or in the field are of major corporate concern. Most modern thinking now suggests that monies spent on defect avoidance are significantly more effective to the overall health of a producer than monies spent to correct field defects and to assuage customer anger.

Computer categories: For our purposes here, computers are classified into two categories, those that are in-line elements of the manufacturing shop processes and those which are used to provide quality program management information and control. The first type noted are often process special to in-line testing and may not always be economically possible for smaller companies so we will not dwell on them here. Rather, we will discuss the use of personal computers for managerial quality analysis and control. These are readily available and within the financial ability of virtually all. Since the scope of quality assurance is so broad, it is useful to confine our examination to the first three issues listed above: pre-production analysis, product/process planning and purchased materials control. The remaining four topics would require extensive coverage reaching beyond the range of this paper.

Pre-production analysis: Perhaps the first step in the establishment of a quality assurance program, after initial organization, is that of pre-production product analysis. It is the first logical step in the sequence of the

product development process and it is at this very early point that quality considerations can and should be introduced. Introducing quality conceptualizations at the earliest possible time also contributes towards establishing an on-going atmosphere of quality awareness throughout the manufacturing cycle. Each business may have its own approach to executing pre-production planning, but for most products the following program is often appropriate: (1) study of program feasibility - after marketing viability has been determined (2) design -- product & process (3) pre-production trial run(s) (4) production start-up (5) production quality continuity.

Program feasibility concerns itself with such elements as preliminary design parameters, product specifications, piece-part sub-assembly and assembly relationships, choice of materials and finishes and engineering tolerances. Each of these lends itself to the use of various computer analysis. For small companies, much of this can be obtained from company historical records and maintained on disks. For those still maintained in manual systems only, the information can be transferred to disk files.

Pre-production trial runs: For sophisticated manufactures, the pre-production trial run is a stage between that of the engineering test models and prototypes and actual "all up" production of product for sale. It is a time for debugging both the design and the manufacturing process. It should be a time to consider the introduction into the production process of appropriate inspection and test considerations. Normally, these would include determination of inspection and test points in the process as well as the nature of the inspections and tests to be performed. As part of these activities, it is important that the quality control reporting system be installed and tested. PC computer applications can be introduced here to assure that the reporting of defects and failures is done in such a way as to provide managers with timely and accurate data.

Production start-up: Production start up includes such items as final debugging of the manufacturing processes and observing the behavior of the product during the start-up period for material suitability. It also includes identification of those factors which have an effect on product quality. Selecting levels of acceptability for various quality factors, performing tolerance measurements to assure that product life will be understood especially in terms of strength, stress, reliability and appearance are done at this time. All of this is to assure that the end results of the agreed upon production process will provide a product which meets the levels deemed necessary to compete in the market-place. Each of these activities is directly adaptable to personal computer disk files for storage and retrieval as needed at each stage of the manufacturing process.

Marketing role: It is clear that the use of informational computing is of major importance in processing activities. Data must be obtained, recorded, compared, analyzed and acted upon. In most businesses, the marketing component has an interest in the results of this work and marketing will usually want to have input into the production process. Much of the activity described above can be provided through computer printout to the marketing function so that the manufacturing end results meet the requirements of consumers.

Review and feedback: Production quality continuity requires vigilance to assure that agreed upon design is maintained during the manufacturing process. Review and feedback on material is required from receipt through storage, processing, assembly and shipment. Systems are necessary to identify and control factors which impinge upon

operator performance, transportation and storage of the product, machine controlled practices, and any environmental conditions which might be critical to the final performance of the product. At the point where each of these activities is performed, a personal computer can be stationed and in addition to its application to these processing issues, this same PC can be used to detail, sort, classify and summarize inspection and test data.

Product-process planning: Quality surveillance is an important part of any effort to maintain production of acceptable products. Unfortunately and precisely because it is an after the fact occurrence, it is a cost additive activity. Additionally, since detection of defects further along in the manufacturing process is usually more expensive to correct, it is incumbent upon the small and medium sized business to learn to minimize their occurrence. Rather than rely upon quality surveillance alone then it is preferable for a business to seek quality assurance through the use of controls which minimize the occurrence of inspection defects in the first place. This is a major objective of product-process planning.

Control: Control refers to the processes which generate and measure product characteristics as established by the desired and pre-set levels determined to meet the competitive requirements of the market place. To obtain control there are some 15 elements which must be carefully considered: (1) Establishment of control points (2) Determination of desired characteristics (3) Desired quality levels (realistic) (4) Review of process capabilities (5) Development of control procedures - a) personal selection b) training c) operator involvement d) quality information tools e) how to check during process f) when to check/frequency g) type of sample: Random/consecutive (6) Defect disposition routines/record keeping (7) Routing procedures (8) Quality equipment calibration plan (9) Maintenance plan for quality equipment (10) In-process instpection and test (11) Final test/inspection and acceptance (12) Quality auditing plans for outgoing product (14) Establishment of the quality data feedback system (15) Determination of the measurements and methods to assure the effectiveness of the control plan.

Information now achievable: The decade of the 1980s has seen the introduction of microcomputers from simple machines to full-blown workstations that have the speed and power of older minicomputers. With this type of machine, quality assurance can be upgraded to a great degree. Companies do keep information including time reports, status reports of problems and logs from operation systems. Most of these are kept as informal records, but, with the use of a microcomputer and the proper software programs, this information can become a vital part of quality assurance. The historical data from status reports of problems indicate a natural place for a product evaluation and control. This historical data may include vendor parts that have been used in production up to this point in the process. Specific deficiencies can indicate a lack of adequate inspection at delivery or a deficiency somewhere in the process. Several software programs can be installed that would allow for separate measurements of equipment and maintenance of equipment, materials, and design of the process capabilities. The information obtained by these measurements can be combined to isolate the cause of a production problem or to indicate that several factors together combine to cause the problem. Faulty design may be the lone factor or it may combine with the fact that skill levels of the workforce are inadequate. The problem can also arise from equipment worn out or not properly maintained, or from improper materials received from vendors. The collection of this information allows for analysis of the

product and the process and indicates inspection points for control of both the process and the product. Standards can be set in terms of measurable quantitative data and samples can be tested against limits needed to insure proper production. The same procedure can be introduced during the process where other reports of problems have occurred. Standards can be set so that economically available software products for sampling either indicate rejection of the product or that required reworking is needed.

Quality auditing: Quality auditing is a way of measuring product quality on a selected sample size, frequently selected over a period of time and at predetermined intervals of time. These intervals can be changed to further randomize the sampling process. The quality should be an independent audit and is not meant to replace standard sampling plans which may be used during the normal operations of the inspection function and which are easily available through any number of commercial software programs. The product quality audit is a measurement of the effectiveness of the quality assurance system. Thus, it needs to establish some kind of index or rating system by which the entire system can be evaluated as to whether or not it is improving, deteriorating or staying at the pre-planned levels. All of these have important cost ramifications. For instance, if an overall outgoing quality level of say 97.5% is established as a competitive need of the business, then all contributing departments should be measured against this standard.

The simple example in Table 1 suggests that quality controls in four departments need some fine tuning to bring performance up to the required level. In the sub-assembly operation there is a clear indication that significant action is required to reduce the number of rejects. Such information can be retained daily or weekly depending on the product cycle and is easy to transfer to a computer. With this base number available, supervisors can readily determine needed levels of quality improvement.

Purchased material quality control and organization: Here, we are not only talking about the purchasing function of procuring materials and services for a business, but also the activities of receiving and stocking at economical quality levels, materials which are in conformance with predetermined levels. Most often, purchased materials are obtained in two ways: 1) from outside vendors 2) from other divisions or departments of the company. Either method should result in the procurement of materials which conform to specifications needed to meet the requirements of manufacturing and assembly operations. To assure that this happens, it is useful to consider organizing the quality function into three areas for appropriate coverage: 1) Quality engineering to assure in-coming materials quality specifications 2) Inspection and test to provide incoming quality assurance 3) Process control to monitor and insure adherence.

Such organization is most effective when it is bound to the purchasing component with a set of practices and procedures which clearly define the quality responsibilities of the buyers and vendors. Some basics included in this area: 1) Establishment of a vendor rating plan. 2) Development of a vendor certification plan which would tell the vendor: a) quality requirements b) the acceptance procedures in-house, including sampling plans c) vendor quality data required in advance for review d) how vendor data will be compared to in-coming inspection findings e) proposed actions in the event of vendor failure to meet the in-coming quality standards. 3) The effectivity of the company's buyers in matters of quality needs also to be evaluated: a) How are each buyer's vendors doing in quality performance? b) What are the nature and frequencies of

quality problems attributable to their vendors over time? c) What is each buyer doing about quality problems? The elements of vendor certification and buyer's purchasing lend themselves to quantification and management analysis through the use of informational computers.

COMPUTER APPLICATIONS

Several practical uses of the computer in the quality assurance process can gradually be introduced to improve, and, possibly increase production. Whether materials used in the process come from outside vendors or from other departments inside the organization, a database can be setup to keep track of the ratings of the materials that cause problems. This database should include a rating for the initial inspection of material and for quality control points throughout the whole process. A program may then be written to report on the quality of materials and to rate vendor performance. Commercially available computer programs can also be used to set up sampling procedures. Random sampling can be set up at any stage for in-process inspection and, random sampling tables can be used to indicate that a production lot is either good or bad. Several measurable characteristics, such as size, weight and length, can be entered into the computer to check against performance standards. If the random sampling indicates problems, then several choices can be considered. The first might be to increase the amount of sampling to see if the problem continues through increased sampling, and, in this case, reject the lot. If the increased sampling indicates only some variance from the standards set for a good lot, then the process may have to be sent back for reworking. Historical data generated from in-process inspection can eventually change the norms for size, weight, and length and be reentered into the orginal program, usually resulting in cost savings or cost avoidances. The cost of computer hardware and the increased power available for the quality assurance process is no longer a luxury, but has become a necessity. The introduction of a personal computer for each worker at a predesignated inspection point should eventually be considered. These workers can be taught to run programs that test for the quality of the product at these stations. Both the manual process and the computer process of control should run concurrently until the computer program norms are brought to a satisfactory level.

Receiving and stocking: Just as the pre-production quality actions of Purchasing will dramatically effect the product quality picture, so too can pre-production planning to physically receive, segregate and stock materials in a quality "smart" way. Accordingly, there are several considerations which should be made to provide a coherent control system. These, as a minimum will include: 1) Establishment of incoming quality levels by material types. 2) Provision for specific acceptance plans and procedures. 3) Development of disposition or rejection of questionable materials routines. 4) Establishment of material routing procedures. 5) Development of control procedures to insure that the above are adhered to.

All of this material is easily stored on a PC computer and with the use of networking can be made available throughout the pre-production phases of manufacturing activities. One bit of serendipity arising from such a system is the value to the Production Control function that flows from knowing the up-to-date status of all materials and the ready availability and location of quality accepted materials.

Good quality no accident: In the final analysis, consistent quality performance will occur through the use of a complete quality assurance program. Good quality is rarely accidental and only those companies zealous about obtaining and maintaining competitive quality

standards consistently achieve it. There is more to be done than we have described in the pre-production stages but what has been described above represents a firm initial start for small and medium sized businesses. Through the use of relatively inexpensive quality management informational systems, the effective marriage of low cost computers and quality assurance techniques to all enterprises is possible. For the small-to-medium sized enterprise engaged in competition with financially stronger domestic and foreign corporations, a low cost computer assisted comprehensive quality assurance program is a powerful tool.

BIBLIOGRAPHY

Browne, Jimmie, Harhen, John Shivnan, James Production Management Systems, Addison-Wesley, 1988.

Gibson, Michael Lucas "Implementing the Promise", Datamation, Vol. 35, No. 3, Feb., 1989 pp. 65-68.

McClure, Carma "Methodology in Path From Art to Science", Software Magazine, Vol. 9, No. 7, June 1989, pp. 33-42.

Necco, Charles R. Tsai, Nancy W. and Halgeson, Kreg W. "Current Usage of Case Software", Journal of Systems Management, Vol. 40, No. 5, May 1989, pp. 6-11.

Tom, Paul Managing Information As a Corporate Resource, Scott, Foresman and Company, 1987.

Table 1. Example: For a Selected Time Period

Department	Total Parts Processed	No. of Rejections	% Accepted
Drill Press	10,000	270	97.30
Milling Mach.	10,000	320	96.80
Welding	6,000	154	97.44
Sub-assembly	5,000	300	94.00
Final-assembly	2,500	63	97.48

INTELLECTIVE SKILLS AND INFORMATION SYSTEMS EDUCATION

Paul A. Dorsey
Millikin University

ABSTRACT

In In the Age of the Smart Machine, Shoshana Zuboff clarifies the "informating power of computer technology" in the workplace and the intellective skills required to take advantage of this "informating power." This paper clarifies some educational implications of Zuboff's book and major thesis. The focus of this clarification is on implications of information systems education within an undergraduate business school setting. Educational practices to promote the development of intellective skills, both for prospective information systems end users and development professionals, are introduced and clarified.

INTRODUCTION

An important challenge facing information systems educators today is to help prepare students with the skills required for the changing workplace. An important dimension of that changing workplace is the ubiquity of computers and the increasing volume of informational by-products of computing systems. The most ambitious attempt to clarify the social and organizational implications of these changes is presented in Zuboff's [6] In the Age of the Smart Machine: The Future of Work and Power. Zuboff's analysis helps to clarify both the choices facing our organizations and the skills required for work in those changing organizations.

Zuboff's analysis of information technology in the workplace centers around an insightful understanding of the "dilemmas of transformation" in the age of the smart machine. The major dilemmas posed revolve around the two faces of "intelligent technology". One face of information technology makes it possible to rationalize work activity by making explicit the worker's knowledge and actions and encoding that knowledge and action into the computing system. The effect of this face of information technology is to de-skill work through automation, by taking the worker's knowledge and encoding it into the computer program. Many of the more pessimistic accounts of workplace computerization focus on this face of information technology.

The second face of information technology distinguishes current information technologies from earlier machine technologies. Information technologies, by translating information into actions, also keep track of data about those automated activities and potentially generate new streams of information for workers and decision-makers. This second face of information technology Zuboff terms the "informating" capability. The opportunities and possibilities associated with this "informating" face of computing technology are central to Zuboff's understanding of the opportunities in the workplace created by the smart machine.

The concept of "intellective skills" is introduced and discussed by Zuboff to clarify the new skills required for work in this computer-mediated environment. Important problems and opportunities arise in these environments, and workers who lack the capability of understanding systems (e.g., based on the readings of gauges or the outputs of reports) are unable to deal with these problems and opportunities of the computer-mediated

environment. The tasks of solving problems and seizing and exploiting opportunities require interpretation, the formation and testing of hypotheses, collective thinking and problem-solving, and continual learning. The task for workers is the construction of meaning from the abstraction of an electronic text that represents some structure or process (e.g., the production process in a paper mill, the processing of information in the insurance industry). "Computer literacy" in such a context "means knowing how to look for information, how to gain access to it, and how to do something interesting with it when you've got it [3:58]." The skills required for the "informated" environment are related to the kind of "explicit, inferential, scientific reasoning traditionally associated with formal education [6:195]." Zuboff contrasts these intellective skills with the acting-on skills that are associated with more conventional blue-collar and lower white-collar tasks.

Zuboff's analysis of the smart machine is framed in the context of a sophisticated historical and sociological analysis of social change and is based on a series of ten case studies of information technology and organizational change. The case studies focus on operational tasks in the factory and service sectors. One of Zuboff's major arguments, based on her case studies, is that information technology opens up opportunities for creative use of the "informating" potential of new information technologies, but that pressures by managers for greater control often frustrate those possibilities. This results, according to Zuboff, in the failure to take advantage of new "problem-solving" opportunities and of new information-based products and services which could be created in this new environment.

Zuboff's analysis is important to educators in several respects. First, it frames an understanding of information technology and change that helps us make sense of current developments in the workplace. Second, by clarifying the nature of that changing workplace, it helps to clarify the choices that information systems educators and their students face as organizations are built and modified to exploit the potential of the smart machine. Finally, it clarifies some of the educational implications of these changes with the concept of "intellective" skills, skills which are necessary if we are to exploit the "informating" potential of computer technologies. In the discussion below, for the sake of argument, Zuboff's conclusions about exploiting the informating potential of new work environments are accepted. It is assumed that it is desirable to create new workplace environments that require the intellective skills that are the focus of this paper. This paper focuses on the educational implications of her argument.

INTELLECTIVE SKILLS AND INFORMATION SYSTEMS EDUCATION

It is this concept of intellective skills as it relates to the tasks of information systems educators that is the focus of this paper. More specifically, the focus here is on information systems education in an undergraduate business school environment committed both to end user education and to the education of information systems professionals. Zuboff begins the task of clarifying this concept, especially with reference to the case studies cited in her text where workers require intellective skills to interpret information and to take appropriate action. It is interesting that the new skills required of the workers in the paper mill provide Zuboff with her best materials to illustrate these skills. Since the case studies in Zuboff's book focus primarily on the operational tasks of blue-collar and white-collar workers, it may not appear that they have direct relevance

to a business school setting where students are being prepared for management roles. However, management roles and MIS roles are being impacted by these same changes, just with different kinds of information (less direct, more summarized) available in the information-intensive environment affecting them.

The question should then be posed for the information systems educator in the business school setting: What is the relevance of Zuboff's understanding of the changing workplace for the information systems curriculum? What kind of curriculum can nourish the development of the intellective skills necessary to act constructively in this new workplace? What types of educational programs and projects should we create to help to build those skills and prepare students to work in these new, information-intensive work environments?

An answer to those questions is probably best divided into separate discussions of education for (1) end users of computing and for (2) the more traditional information systems students being prepared for roles in systems development and systems management. This paper suggests that intellective skills are appropriate for both types of students. However, "data" and "information" for end users and information systems professionals are different, and the technologies required for re-creating real-world type environments for end users and information systems professionals are also different.

INTELLECTIVE SKILLS AND END USER COMPUTING

An environment conducive to the development of intellective skills for business end users would seem to require a combination of access to data (e.g., the by-products of information systems) and access to modelling tools for making sense of that data. An environment for the development of intellective skills would be a decision support environment with databases providing the information, modelling tools (spreadsheets, statistical software, project management software, etc.) providing additional means of analyzing that data, and a user interface to the data and modelling tools.

When one thinks of a business school course that would help to develop intellective skills, the capstone course in Business Policy comes to mind. The Business Policy course is generally designed to help the student to develop an understanding of the interaction of all of the functional areas of business. This is accomplished through such instructional tools as business cases and simulations. The standard business case includes both qualitative information (in the form of the general case write-up) and quantitative information (in the form of balance sheets and income statements) that the student uses in developing an "abstract" understanding of the case, on which are based analyses, recommendations, etc. The standard business simulation, a computer-based instructional tool, implements an abstract model of a business reality and requires students to make business decisions to promote the financial well-being of their firm in that abstracted business reality. Summarized information, a by-product of the computer simulation, is provided to students to help them understand the consequences of their business decisions and, also, to understand the more abstract business "reality" which determines the success or failure of their decisions. Both business cases and business simulations can promote intellective skills, though the interactiveness and feedback provided in a good simulation are probably more likely to be successful.

Within our business school, we have developed an information systems integration program that is designed to promote those intellective skills in

more specific functional area courses. One aspect of this program has been to work with local corporations to develop information systems for curricular use based on "real-world" data that can be incorporated into our business school curriculum. Three examples, which are projects completed or underway, illustrate the kind of system that has been the focus of our activity.

-A marketing information system, based on insurance area data--demographic, policy, etc.--which has been used in a marketing management class to help develop analysis and decision-making skills. The data is organized in a Lotus spreadsheet with macros based on data management commands to provide summary data to the user.

-An insurance policy retention information system, again based on insurance agency data, which has been used in a decision support systems class to help develop analysis and decision-making skills. The focus is on identifying client, agency, geographic, and other factors which may contribute differentially to poor retention and on developing policy recommendations (based on that data and other company data provided to the student in the form of "case" information).

-A cost accounting information system, being developed based on data provided by a company highly committed to manufacturing automation. The information system will be used in the advanced cost accounting class to help students to make sense of, and develop policies for, handling the allocation of costs in the new manufacturing environment.

For each system, an environment is created which consists of real world data--organized for access within a software package with data management facilities, tools for analyzing that data, and with real world problem/decision situations.

Working with businesses to develop information systems which can serve curricular goals is a time-consuming task. We have used these opportunities as a means of getting honors students involved in real world projects. The tasks of tailoring the project surrounding the information system so that it has enough structure so that students don't flounder and not too much structure so that students are posed with a genuine challenge is a difficult one. But our experience suggests the promise of this approach. We have pinpointed more advanced business courses (e.g., marketing management, training and development, advanced cost accounting) as the curricular "location" where these curricular innovations designed to promote intellective skill development are most appropriate.

INTELLECTIVE SKILLS AND MIS EDUCATION

If, for end users, business databases with modelling tools used on challenging business problems constitute the basis for an educational environment for the development of intellective skill, what is the corollary of this type of environment for students majoring in Management Information Systems (or Computer Information Systems)? On first thought, the answer to that question is not self-evident. However, reflection on that question, coupled with an understanding of current developments in MIS, leads one to the clear conclusion that the emerging CASE (Computer-Assisted Software Engineering) environment is the information systems professionals' information-intensive environment. Unlike the area of end user information systems education, at our institution we have done relatively little in making sense of how to create a new, more challenging and realistic environment for information systems education based on the creation of a CASE environment. And the current technology is an obstacle, both in terms

of cost and integration of CASE tools.

CASE involves the application of information technology to the role of the systems development professional. CASE tools begin by "automating" certain systems development tasks. However, information about systems (specifications, diagrams, etc.) which results from the activities of the professionals using CASE tools is the information by-product of the systems development effort. This illustrates the "informating" potential of CASE, argued for in a recent review of the present status and future directions of computer-aided software engineering [2]. This "informating" potential can and should have the effect of improving the quality of information systems end products through better "information" use (e.g., improved communication). The recent focus of CASE efforts has been on developing an integrated "repository", where all object definitions and relationships are stored. Such a repository is generally seen as the axis around which CASE pivots.

The discussion of CASE in the MIS literature largely focuses on CASE as a tool for "automating" the systems development life cycle. Productivity gains for systems professionals become the major focus of CASE efforts and the major rationale for justifying CASE developments within organization [5]. The problem becomes one of identifying the magnitude of "productivity" gains necessary to justify the purchase of CASE tools and other CASE-related costs (training, etc.). However, there is still controversy surrounding the claims of improved productivity from the use of CASE tools [4]. Also, qualitative dimensions which are difficult to translate into hard productivity data (e.g., quality of documentation and systems design and improved communication and team-building) are frequently cited as benefits of CASE.

If CASE technology provides the "informating" potential of an educational environment to promote the development of "intellective skills", what educational implications can be drawn from that understanding?. CASE tools are being gradually introduced into the curriculum, though the rationale for the introduction and use of these tools is not precisely clear. Certain CASE tools, such as code generators, may be used to make it possible to complete information systems projects within one course, by "automating" certain systems development activities [1]. Some CASE tools may be introduced to prepare students for work environments that use that specific tool. CASE tools may also be used to help reinforce the use of structured techniques for systems development.

While these uses of CASE tools may be laudable, there is a more ambitious approach which forward-looking information systems educators should be considering as they evaluate CASE as a set of technologies and as an approach to systems development. Just as data-rich environments can be developed for business end users (e.g, marketing or personnel students), so an environment rich with information about pre-existing information systems (a CASE information environment) can be provided for budding information systems professionals. That environment can provide the context for student systems development work. Some of that development work can be on new systems, but which use information (specifications, etc.) already stored within the system. Projects that span courses with different students in those courses can be considered with the CASE environment providing an important link. Projects can be developed requiring the maintenance of systems for which information is already stored in a CASE repository. Maintenance work that is too frequently neglected in the MIS curriculum might even insinuate itself into the MIS curriculum. These are just a few of the kinds of projects students could work on in such an application development environment.

Intellective skills require abstraction, the making explicit of that knowledge to facilitate its use in hypothesis formation and problem-solving. CASE as an approach requires the making explicit of systems knowledge and provides access to that knowledge to those with access to the system. Student work in a CASE-type environment, where project challenges require the student to exploit the capabilities of that system, should help to promote these higher order skills. Certainly the skills developed in standard project work in information systems courses should promote solid problem-solving skills. But there would seem to be advantages to trying to simulate a more complex and realistic information systems development environment which would provide a range of challenges for students with a range of capabilities and motivations.

Clearly the current state of CASE technology, coupled with the costs of CASE tools, constitute powerful obstacles to a realization of the vision presented above. But the maturation of the technology, with increased agreement on standards, and the emerging reality of the OS/2 environment provide promise for improvements in the technological underpinning of a new information systems curricular ideal.

CONCLUSION

This paper has explored some of the educational implications of ideas presented by Zuboff in In the Age of the Smart Machine. The ideas presented in that book provide an opportunity and challenge for information systems educators to play a central role in preparing students for emerging workplaces. The approaches and ideas discussed in this paper are an attempt to initiate a dialogue among information systems educators about how we can prepare students and organizations to exploit the "informating" possibilities of new information technologies.

REFERENCES

(1) Amadio, William J., "Systems Courses in the 1990s: The Promise of CASE and 4GLs", Interface, Fall 1988, pp. 14-18.

(2) Chen, Minder; Nunamaker, Jay F.; and Weber E. Sue, "Computer-Aided Software Engineering: Present Status and Future Directions," Data Base, Spring 1989, pp. 7-13.

(3) Emmons, Garry, "Smart Machines and Learning People: An Interview with Shoshana Zuboff," Harvard Magazine, Nov-Dec. 1988, pp. 56-60.

(4) Norman, R.J. and Nunamaker, J.F., Jr., "Integrated Development Environment: Technological and Behavior Productivity Perceptions," Proceedings of the 22nd Hawaii International Conference on System Sciences, January 1989.

(5) Santonsus, Cathleen, "The fine art of figuring CASE payback," Computerworld, March 27, 1989, pp. 74-75

(6) Zuboff, S. In the Age of the Smart Machine: The Future of Work and Power, New York, Basic, 1988.

Computer Aided Software Engineering

David Wallace
Illinois State University

Scheyla Vasconcelos
Brasilia, Brazil

ABSTRACT

Computer Aided Software Engineering (CASE) is a powerful information system management and development tool. Describing CASE tools to management, data processing personnel, or any user group within an organization would cover the components/functions of CASE, the CASE market, the evaluation and selection of CASE tools, the difficulties/benefits of CASE tools, an its future. CASE tools are separated into two major categories: (1) front-end analysis (analysis and design), and (2) back-end tools such as code generators, programming, testing, and maintenance. The CASE market is a very lucrative and competitive market with many organizations introducing their products or modifying existing product lines. Consequently, users need to establish a formal evaluation and selection process to identify the CASE product best suited for their needs. They need to realize the benefits/difficulties which are associated with CASE tools in order to make their selection process effective. The benefits fall into two general categories: (1) improvement of productivity and quality in the information system development process, and (2) the integration of the organization's information systems for management and strategic direction. The difficulties are limited capacity in terms of size and complexity, poor interface integration, and the difficulty of changing from a programming environment to a software engineering environment. When advances in technology and standardization are realized in the CASE field, organizations will be able to model their entire information systems for greater communication, integration, and strategic direction.

Interface Issues in Building Applications in a Database Environment

Chang-Yang Lin
Engming Lin
Eastern Kentucky University

ABSTRACT

This paper demonstrates the importance of query language as an interface between an application program and the database, Two application programs are used to illustrate this procedure. One is a frames-driven program written in a relational language where query language statements are used to link frames with the database. The other one is an embedded COBOL program where query language statements perform preliminary input/output tasks. By illustrating the examples, interface issues such as query languages, frames, operation specification languages are identified. The implications of these interfaces to the systems development process are discussed.

Managing Information Technology at Cincinnati Financial Corporation

Catherine Bishop-Clark
Miami University, Middletown

Rebecca Barens Koop
University of Cincinnati

ABSTRACT

In order to survive in today's competitive marketplace, many companies have shifted the orientation of their information technology management from traditional to strategic applications. The current literature most often cites only the few well known examples of strategic applications, such as American Hospital Supply. But effective management of information technology has led to strategic applications in other companies as well. This paper discusses several of the relevant issues concerning strategic management of information technology using the Cincinnati Financial Corporation to highlight the effective integration of theory and practice.