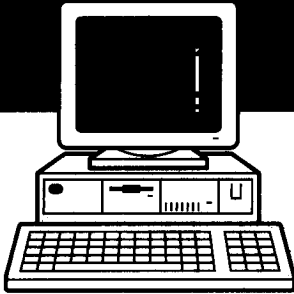


# ISECON '93

## Information Systems Education Conference



### IS Education The Future of Information Systems

November 5 - 7, 1993

Phoenix Civic Plaza

Phoenix, Arizona

### Proceedings of the Tenth Information Systems Education Conference



SPONSORED BY  
DPMA EDUCATION FOUNDATION

Conference Proceedings

## ISECON '93

### *IS Education - The Future Of Information Systems*

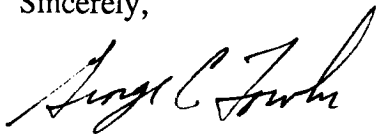
On behalf of the Board of Regents of the Education Foundation of DPMA, DPMA, the Executive Committee and Track Chairs, I thank you for your participation and support of ISECON '93.

The theme of this year's conference is *IS Education - The Future of Information Systems*. I believe the theme would appropriately describe every ISECON conference and even our profession. In a dynamic and ever changing profession like ours, education is the only way to insure the survival of the profession. As IS educators we are integral to the survival and the development of the Information Systems profession. Your involvement in research into curriculum issues and IS teaching methodologies is an absolute must.

Because of your efforts, these proceedings contain many quality papers that should help each of us contribute to *The Future of Information Systems*. These papers reflect many hours of dedicated work by the authors. The papers included in these proceedings passed a peer review process and I thank the reviewers for all their hours of work and dedications.

Your participation in ISECON, the premier conference in IS education, speaks loudly to the role you perceive ISECON plays in the future of Information Systems. I encourage each of you to get involved in the many sessions provided for you. Share your ideas, research and experiences with your colleagues. Also find time to enjoy some camaraderie and the wonderful host city of Phoenix.

Sincerely,



George C. Fowler, PH.D.  
Conference Chair



Data Processing Management Association

## Education Foundation

505 Busse Highway, Park Ridge, IL 60068-3191

(708) 825-8124

Dedicated to enhancing educational opportunities  
for information systems professionals

---

Dear Information Systems Educators,

On behalf of the DPMA Education Foundation Board of Regents, it is with great pleasure that I welcome you to ISECON '93 here in Phoenix. This is an excellent opportunity to learn the latest in thinking among the nations top educators.

The conference committee, under the chairmanship of Dr. George Fowler, has developed an outstanding conference for you. The educational programs are among the best offered anywhere and we hope that you will take full advantage of them.

Speaking of taking advantage of things, take time to enjoy our wonderful Arizona weather. Visit the many wonders of our state. Don't miss the fine shops, restaurants and museums within easy walking distance of the Convention Center.

Be sure that you attend the Saturday luncheon where the DPMA ED-Sig/DPMA-EF Educator of the Year award is presented. This is always an inspiring event and I'm sure that the winner of this years award will have some interesting things to say.

All of the elements of a superb educational experience are here. An outstanding keynoter, sessions on the latest in Information Systems Education, awards presentations, a beautiful, active city and great weather. The only thing that is left up to you is to participate!

Welcome to ISECON '93.

Best Wishes,

Gilbert R. Hedger  
President,  
DPMA Education Foundation  
Board of Regents

# ISECON '93 STEERING COMMITTEE

## **DPMA EDUCATION FOUNDATION PRESIDENT**

Gilbert (Gil) Hedger, CSP  
Disaster Contingency Planning  
Scottsdale, Arizona

## **CONFERENCE CHAIR**

Dr. George Fowler  
Texas A & M University  
College Station, Texas

## **PROGRAM CHAIR**

Dr. Alden Lorents  
Northern Arizona University  
Flagstaff, Arizona

## **PROCEEDINGS CHAIR**

Dr. Herbert Rebhun  
University of Houston-Downtown  
Houston, Texas

## **PLACEMENT CHAIR**

Dr. James Trumbley  
University of Texas-El Paso  
El Paso, Texas

## **EXHIBITS CHAIR**

Dr. Mary Sumner  
Southern Illinois University-Edwardsville  
Edwardsville, Illinois

## **LOCAL ARRANGEMENTS CHAIR**

Dr. Jane Carey  
Arizona State University-West  
Phoenix, Arizona

## **EX-OFFICIO, ISECON'92 CHAIR**

Dr. R. Wayne Headrick  
New Mexico State University  
Las Cruces, New Mexico



## **TRACK CHAIRS**

### **IS CURRICULUM**

Dr. Eli Cohen  
Eastern New Mexico University  
Portales, New Mexico

### **END-USER COMPUTING**

Dr. Karen Forcht  
James Madison University  
Harrisonburg, Virginia

### **EMERGING TECHNOLOGIES**

Dr. Tom Sandman  
California State University-Sacramento  
Sacramento, California

### **CIS PROGRAM DIRECTIONS AND ISSUES**

Dr. Carol Chrisman  
Illinois State University  
Normal, Illinois

### **HUMAN FACTORS**

Dr. Carol Saunders  
Florida Atlantic University  
Boca Raton, Florida

## ISECON 93 PAPER REVIEWERS

A special thanks to the following professors who reviewed papers for the 1993 Information Systems Education Conference:

**Russell Ching**  
University Arkansas

**Betty Kleen**  
Nicholls State University

**Claude Simpson**  
Northwestern State

**Kalmesh Mehta**  
St. Mary's University

**James Nelson**  
New Mexico St. University

**Joseph Williams**  
Colorado State

**Thomas Hilton**  
Utah State

**Ron Kizior**  
Loyola University Chicago

**Mary Jo Haught**  
MJ Systems

**Kathryn McCubbin**  
Christopher Newport U.

**Warren Duclos**  
Tulane University

**Dan Radell**  
Winston-Salem State U.

**Bruce White**  
Dakota State University

**Larry Cornwell**  
Bradley University

**Billy Lim**  
Illinois State University

**Luann Stemler**  
Illinois State University

**Gerry Chrisman**  
Illinois State University

**Krishnan Muthuswamy**  
Illinois State University

**Robert Rariden**  
Illinois State University

**Karin Bast**  
University of Wisconsin

**Karen Forcht**  
James Madison University

**Heidi Owens**  
Portland State University

**Chin Kuo**  
Eastern Washington U.

**Morag Stewart**  
Eastern Washington U.

**Kathy Moffitt**  
California St. U. Fresno

**Esther Guthery**  
American Graduate School  
of International Business

**Krishnan Muthuswamy**  
Illinois State University

**Kathy Moser**  
Iowa State University

**Shiela Jacobs**  
Oakland University

**Vartan Safari**  
Winona State University

**Shashi Shah**  
Seton Hall University

**Dorothy Dologite**  
CUNY Branch College

**Gerry Evans**  
U. of Montana, Missoula

**Ed Christenson**  
Calif. St. U., Sacramento

**Bernadette Szajna**  
Texas Christian University

**Jane Carey**  
Arizona State University

**Wayne Spence**  
University of North Texas

**Jack Becker**  
University of North Texas

**Fred Harold**  
Florida Atlantic University

**Shaila Miranda**  
Florida Atlantic University

**Carol Saunders**  
Florida Atlantic University

**Stan Revesman**  
Florida Atlantic University

**Jane Mackay**  
Texas Christian University

**Tim Peterson**  
University of Tulsa

# ISECON 1993

## INDEX

### FRIDAY NOVEMBER 5

#### WORKSHOP # 1

CASE Tools for the 90's . . . . . 1

#### WORKSHOP # 2

The Modern COBOL Language--1993 . . . . . 2

#### WORKSHOP # 3

Developing Database Applications Using the Semantic  
Object Model . . . . . 3

#### WORKSHOP # 4

Introduction to UNIX for IS Educators . . . . . 4

### SATURDAY NOVEMBER 6

#### SESSION 1---10:00-11:15

Managing Application Development-Introducing Project . 5-12  
Management Into the Applications Development Cluster  
by Roy Daigle  
University of South Alabama

Applications Development in Client-Server . . . . . 13-17  
Environment: A Step in the Right Direction for CIS  
Curriculum  
by Mayur Mehta and George Morgan  
Southwest Texas State University

Better End-Users--IS Design Issues for the MIS Class 18-23  
by Kathleen Mofitt  
California State University--Fresno

Partnering IS with the Business Community: . . . . . 24  
The Information Systems Research Institute  
by Jean Gasen and Edwin Blanks  
Virginia Commonwealth University

Educating the Business Educator-Faculty Internships . 25-30  
by Thomas Loughman and Robert Fleck  
Columbus College, Georgia

The Impact of CASE Tools on Achieving Critical Success Factors in Systems Development by Mary Sumner Southern Illinois University--Edwardsville	. . . 31-39
Data Communications: A Look at Content Revision by Gerald Tatar Duquesne University	. . . 40-41
Integrating Information Systems Technology in the Organization of the Future by Herman Hoplin Syracuse University	. . . 42-49
Managing the End-User: An End-User Computer Course for the 90's by Irv Englander Bentley College	. 50-54
Empirical Study of Information Systems Downsizing by Arjan Sadhwani and B.S. Vijayaraman University of Akron and H.V. Ramakrishna, Salisbury State University	. . 55-61
IDST: A GUI-Based Tutoring System for Learning Data Structures by Billy Lin, Jean Wang, Chris Kohlbrecker and Jamie Jason Illinois State University	. 62-69
 <b>SESSION 2--12:30-1:45</b>	
Design Comes Before Programming by John Schrage Southern Illinois University--Edwardsville	. . . . . 70-77
Introducing Teamwork Through Programming Courses by Luann Stemler and Carol Chrisman Illinois State University	. . 78-83
COBOL-85's Impact on Teaching Programming: Opportunities for Improvement by R. Wayne Headrick New Mexico State University	. . . . . 84-88
What Directors of MBA Programs Think About the Structure and Content of Information Courses by H.V. Ramakrishna--Salisbury State University and V.A. Quarstein--Old Dominion University and B.S. Vijayaraman--University of Akron	. . . 89-96
Survey of MBA-MIS Programs by James Morgan and Craig VanLengen Northern Arizona University	. . . . . 97-103

Integrating IEF into the CSUS Undergraduate MIS Curriculum by Thomas Sandman California State University--Sacramento	104-108
Information Engineering, IEF, and the CIS Curriculum by Alden Lorents and Greg Neal Northern Arizona University	109-115
Factors to be Considered in Adopting I-CASE for Undergraduate Computing Curricula by Claude Simpson and Debasish Banerjee Northwestern State University--Louisiana	116-118
Do As I Say Do: Software Piracy As a Failure of Ethical Will by Diane Miller University of Southern Mississippi	119-124
Information Compilation and Disbursement: Moral Legal and Ethical Considerations by Karen Forcht, Joan Pierson, and Daphyne Thomas James Madison University	125

**SESSION 3--2:15-3:30**

Implementation of the DPMA Four year Model Curriculum at Three Select Institutions Panel Discussion by Eli Cohen--New Mexico University and Larry Eggan--Illinois State University and Mayes Mathews and Kathryn McCubbin Christopher Newport University	126
Where is our Future in Business Schools: Information Technology or Business Processes? by Steven Alter University of San Francisco	127
Integrating Quality Management into the MIS Curriculum by Barbara Dennison Wright State University	128-133
Incorporating a GIS Component in an Introductory Computer Science Course by Carol Hall, Krishna Agarwal and Alfred McKinney Louisiana State University--Shreveport	134-139
Computers: Developing an Interdisciplinary Writing Skills Tool by Marian Sackson and Ilene Siegel-Deutsch Pace University	140

Business Process Reengineering . . . . . 141  
Tutorial  
by Donald Chand  
Bentley College

**SESSION 4---4:00-5:15**

High School Computer Literacy and Performance in . 142-146  
Introduction to CIS Course  
by Craig VanLengen and Jo-Mae Maris  
Northern Arizona University

The Role of the Introductory IS Systems Course in . . . 147  
Recruiting IS Majors  
by Judith Simon and Ronald Wilkes  
Memphis State University

Development of an Information Systems Curriculum . 148-155  
for Non-traditional Students  
James Pick and K.D. Schenk  
University of Redlands

Are We Really Educating IS Graduates With the . . . . 156  
Communications and Problem Solving Skills to the  
Satisfaction of the IS Industry ?  
Panel  
Herbert Longenecker, Jr and David Feinstein  
University of South Alabama  
and Jon Clark and Carl Clavadetscher--Colorado State  
and Eli Cohen--Eastern New Mexico University  
and Mary Sumner--Southern Illinois Univ.-Edwardsville  
and Robert Zant--University of Alabama-Huntsville

A Project-Intensive Introductory Object Oriented . 157-161  
Programming Course  
by Billy Lin  
Illinois State University

Integrating Object Oriented Technology into a . . 162-168  
Database Course using SOD--a Simple Object Database  
by Conrad Chang--University of Illinois  
and Billy Lin--Illinois State University

Integrating Object Orientation in the IS . . . . . 169-176  
Curriculum  
by Les Waguespack  
Bentley College

Group Systems in the CIS/MIS Curriculum . . . . . 177-178  
Panel  
by Evangeline Jacobs--Northern Arizona University  
and Doug Vogel--University of Arizona

**SUNDAY NOVEMBER 7**

**SESSION 1--8:30-9:45**

- The Future of the MIS Course . . . . . 179-184  
Panel  
by Ray McLeod--Texas A&M University  
and James Senn--Georgia State University  
and Richard Hatch--San Diego State University  
and James O'Brien--Northern Arizona University  
and Mary Sumner--Southern Illinois Univ.-Edwardsville
- Professional Practice:Networking with Business . . . 185-192  
Panel  
by Warren Duclos--Tulane University  
and EDSIG Board Members
- Object Oriented:"It Doesn't Have To Be Difficult" . . . 193  
by Sandra Poindexter and R. Bharath  
Northern Michigan University
- Object Orientation:The Loss of Certainty--Lessons . . . 194  
Leaned When Teaching Object Orientation in the CBA IS  
Systems Class  
by Wita Wojtkowski, W.G. Wojtkowski and Emerson Maxson  
Boise State University
- Are Scripting language Suitable for Teaching . . . 195-202  
Object Oriented Programming?  
by Jo-Mae Maris  
Northern Arizona University
- University Information Systems Department Can . . . . . 203  
Successfully Compete in the Commercial IS Training Market  
by Larry Brumbaugh  
Illinois State University
- Is There a Relationship Between Four-Year University . 204  
Faculty Members' Computer Use and Demographic  
Characteristics?  
by Thomas Duff and Patricia Merrier  
University of Minnesota--Duluth

**SESSION 2--10:15-11:30**

- Can Interactive-Audio Televised Instruction Be . . . 205-211  
Effective? A Review of the Literature  
by Eli Cohen  
Eastern New Mexico University
- LAN: Design and Implantation for a Management & . . . 212  
Skills Course  
by Marian Sackson---Pace University

Multimedia and CAI Authoring Systems . . . . .	213-218
by John Sigle	
Louisiana State University-Shreveport	
Strategic Factors in International Information . . . . .	219
Systems: The Global Perspective	
Tutorial	
by Ronald Kizior	
Loyola University	
Integrating Database into an Application and . . . . .	220
Systems Development Sequences for 2nd & 3rd Year	
IS Undergraduates	
Panel Discussion	
by Robert Zant--University of Alabama--Huntsville	
and Roy Daigle--University of South Alabama	
and Herbert Longenecker, Jr.--Univ. of South Alabama	
and Robert Stumpf--California State Polytechnic--Pomona	
and Lavette Teague--California State Polytechnic--Pomona	
Teaching Ethics in IS Courses: Everything You . . . . .	221
Always Wanted to Know But Were Afraid to Ask	
Tutorial:	
by Ernest Kallman and John Grillo	
Bentley College	

**SESSION 3--1:00-2:15**

Evaluation of Current Database Teaching and its . . . . .	222-223
Future Direction	
Panel Discussion	
by Joobin Choobineh--Texas A & M University	
and Jeffery Hoffer--Indiana University	
and David Kroenke--Wall Data Inc., Seattle	
and Sudha Ram--University of Arizona	
and Fred McFadden--University of Colorado	
Curriculum Implementation--The ACM Two Year College . . . . .	224
Experience	
Panel Discussion	
by Karl Klee--Jamestown CC NY	
and Michael Wolf--El Paso CC	
and Suzanne Gill--Mt Royal College, Canada	
and Mary Jo Haught--MJ Services, VA	
and Rod Southworth--Laramie CC WY	
and Tony Mann--Sinclair CC OH	
The Role of Object Orientation in the IS Curriculum . . . . .	225
Panel Discussion	
by Donald Chand and Les Waguespack--Bentley College	
and Timothy Korson--Clemson University	
and Vijay Vaishnavi--Georgia State University	



Total Quality Management in Information Systems: . 226-228  
Issues and Prognosis  
Panel Discussion  
by Don Beaver--Allied Signal, Phoenix  
and John Boughter--Salt River Project, Phoenix  
and Richard Discenza--University of Colorado  
and Neil Jacobs--Northern Arizona University

**SESSION 4--2:30-3:45**

Accreditation Issues for Programs in Information . . . 229  
Systems

Panel Discussion  
by Joyce Currie Little--Towson State University  
and David Feinstein--University of South Alabama  
and Tom Kirk--DeVry Institute  
and John Gorgone--Bentley College

Interactive Abstract of Object Orientation for . . 230-231  
the IS Curriculum

by Les Waguespack  
Bentley College

Achieving Quality: New Roles of the IS Professional . . 232  
in Supporting End-User Computing and Business  
Process ReDesign

Panel Discussion  
by David Kroenke, Ed Altman, and Greg Boon  
Wall Data Inc., Seattle  
and Jon Clark--Colorado State University  
and Herbert Longenecker, Jr.--University of South Alabama

## **CASE Tools for the 90's**

### **Texas Instruments**

The content of this Workshop will include the following topics:

- a. information engineering using IEF;
- b. repository-based information systems reengineering;
- c. business process reengineering;
- d. CASE development directions in the 90's, including client server, GUI interfaces, and object orientation;
- e. information on the academic grant program.

The workshop format will include speakers, videos, online demonstrations, and handouts.

## **The Modern COBOL Language--1993**

**by**

**Jerome Garfunkel, Micro Focus, Inc.**

This workshop explores the current state of the ANSI COBOL 85 language with the Intrinsic Functions Amendment added in 1989. Examples of all the important new structured facilities in the COBOL Standard will be demonstrated. In addition the 42 new intrinsic functions (i.e. Date function, Character/String functions, Financial functions, Numerical Analysis functions, Trigonometric functions, etc.) added to the ANSI COBOL 85 Standard in 1989 will be presented with working examples using the Micro Focus COBOL Workbench. Also included will be a discussion of the ANSI, ISO, and CODASYL COBOL committees that are responsible for the development of the COBOL language. Future enhancements, currently being considered for the upcoming revised COBOL standard (COBOL 00 perhaps) will be presented and feedback will be solicited from attendees for input to the ISO and ANSI COBOL committees. Issues relating to PC application development (i.e. COBOL vs C) will be discussed as well as the controversy surrounding the "birth" on ANSI COBOL 85.

# **Developing Database Applications Using the Semantic Object Model**

David M. Kroenke, Wall Data, Inc.

## **Abstract**

The development of GUI-oriented database applications is too difficult, too time consuming, and too expensive. As a result, many applications that could be developed cannot be cost-justified. Database management systems on the market today require the developer to master not only database modeling and design, but also GUI-interface design and event-oriented programming. A large portion of this work could, and should, be done by generalized, applications generation software. To do this, however, a semantically rich model of the data must be developed.

The semantic object model provides a framework for developing such a model. A semantic object is a semantic description of something of interest to the user. A semantic object contains attributes that can be simple values, formulas, links to other objects, or groups of attributes. Groups can be nested to any level. Subtypes and supertypes are modeled as is multiple inheritance.

Semantic objects are richer in content and context than are the entities of the entity-relationship model. At the 1993 NCEI conference, the semantic object model was presented to a group of sixty-five professors. Eighty percent of those professors found the semantic object model easier to understand than the entity-relationship model. Twenty percent found it equivalent, and no one thought it was more difficult to understand.

This workshop will include a) demonstrations of database application development using Microsoft Access; b) demonstrations of application development using semantic object definitions as the basis for automated creation of databases and database applications; c) description of the semantic object model, including objects, attributes, profiles, and properties; and d) a discussion of the relationship between semantic object modeling and object-oriented programming. A copy of a Windows application that can be used to create semantic object models and generate schemas for Microsoft Access and Borland's Paradox will be given to all participants, courtesy of Wall Data, Inc.

## Introduction to UNIX for IS Educators

Daniel Farkas  
Chair, Information System Department  
Pace University  
Pleasantville, NY 10570  
Internet: farkas@pacevm.dac.pace.edu

**Abstract:** This half-day workshop provides an opportunity for Information Systems professionals and educators to get an understanding of the UNIX operating system and why it has become so popular. The workshop emphasizes a concepts approach which will permit participants to begin using UNIX after taking this seminar. A system for demonstration purposes will be available for students to experiment with some of the concepts presented.

### **Workshop Topics:**

- I. Introduction: History, Philosophy, Versions
- II. Using UNIX: User interface, files, processes
- III. The UNIX Process Model
- IV. The Information Systems Development Environment
- V. Conclusion: UNIX system comparison,  
Relationship to OS/2, DOS; Summary

# **Managing Application Development**

## ***Introducing Project Management into the Applications Development Cluster***

Dr. Roy J. Daigle

Mrs. Janet J. Kemp

School of Computer and Information Sciences

University of South Alabama

### **Abstract**

Over time, the teaching of applications development has evolved into a software engineering based approach. The DPMA IS'90 curriculum is a plan which aids this evolution. In this paper, we describe a method for introducing concepts of project management into the Application Development Cluster according to the IS'90 curriculum.

### **Introduction.**

During the 1991-1992 academic year, the Information Science faculty made a commitment to redesign the IS curriculum according to DPMA IS'90 model curriculum [Longenecker and Feinstein, 1991]. A major component of this redesign is Applications Development. In this paper, we focus on that portion of the Application Development Cluster which pertains to Project Management concepts.

DPMA IS'90. IS'90 is a planning document for an IS curriculum which presents two views of knowledge. One view consists of a collection of seven knowledge clusters within four areas of knowledge content. Each cluster is made up of concepts and activities that are related and, as a whole, define the expected educational experiences for the cluster. Applications Development is one of the knowledge clusters.

Another view of the IS'90 curriculum consists of a distribution of body of knowledge elements throughout the curriculum (defined by the knowledge

clusters and the body of knowledge elements). Using a descriptive language similar to that of Bloom [Longenecker and Feinstein, 1991], the distribution also specifies a *target learning objective* for the interaction of a knowledge element with clusters.

### Project Management in DPMA IS'90.

Project management is a body of knowledge element that has an exit learning level for each component in each knowledge cluster. In Knowledge Cluster D - Application Development, an exit learning level of 1, Awareness, is appropriate for the knowledge unit project planning. At this level, the student will have an awareness of simple systems. This awareness will expand to related concepts in other clusters. In Cluster E - Systems Development, the student will use an awareness of project planning to develop an ability to analyze complex information systems and to design and implement these systems. Knowledge Cluster F - Systems Project will extend this awareness further by asking the student to develop the ability to apply this knowledge to more realistic problems, those beyond the classroom.

## Rationale of the Method.

Past Pedagogy. Analysis of the IS'90 document for requirements of the Application Development Cluster necessitated an examination of previous methods used for teaching that cluster. A search of textbooks for approaches to teaching programming concepts shows progression from a syntax-centered pedagogical approach of the 60's [McCracken, 1963] and early 70's [McCracken, 1970; Philippakis, 1974] to a structured methods approach of the 70's [Feingold, 1978; Philippakis, 1978; Grauer, 1979; Shelly and Cashman, 1977, 1978] and 80's [Feingold and Wolff, 1983, 1988; Grauer, 1981, 1983; Welburn, 1981, 1983]. The structured methods approach has been replaced by a software engineering approach [Janossy, 1989; Uckan, 1992]. Although our understanding of how to better communicate these programming concepts grew simultaneously with improvements in the pedagogy, problems with teaching application development persisted.

Obstacles to Development. There can be many obstacles that impede the ability of students to learn about a topic. Time to develop an application is a major one in the applications development courses. The initial overhead for beginning a new application, the awkwardness of mainframe environments, and the time frame for the class are all time-related factors which can affect development. Some assistance may be provided by taking a *problem class* approach to design [Daigle, 1985]; this means using examples and assignments that are similar to the student application. By encouraging "cloning" of previously developed applications or by using a "phased build" approach in the construction of programming assignments, further obstacles to development can be lessened. The "phased build" method, as described in

[Bergland and Gordon, 1981], employs a project management approach to applications development. The application is developed in stages so that the outcome of each stage is a testable, usable product. [Sigwart, Van Meer, and Hansen, 1990] give a more comprehensive approach to application development which includes a generalization of the "phased build" approach.

Managing the application development process is a time-related problem. Although structured methods and software engineering principles provide guidelines and techniques for the evaluation of an application, **managing application development** is still essentially left to the student to discover.

A Proposed Solution. Introducing Project Management at the Awareness level into the Applications Development Cluster can be achieved by presenting a method for managing application development. This approach would help students contend with obstacles of developing applications.

In the remaining sections of this paper, we

- describe the basis for the incorporation of Project Management concepts into the teaching of management of the development process,
- describe how the introduction has been accomplished in the first course of the cluster,
- describe the reinforcement of the concepts in the second course; (this complies with the spiral learning principle of IS'90 [Longenecker and Feinstein, 1991]) and
- list conclusions and lessons learned from the initial effort.

## Methods for Integration.

Although the IS'90 curriculum provides a framework for identifying *Where* in the curriculum and at *What* learning level a knowledge element is blended into the curriculum, it gives no direction as to *How* to accomplish the interaction. The principal use of student assignments is to contribute *Application* [Bloom, 1956] level learning about course knowledge elements. Assignments may also serve as a way to reinforce current knowledge elements or to introduce new ones. The use of assignments in this manner is one vehicle for implementation of IS'90.

After a brief introduction of terminology, our method has been to illustrate the concepts of project management by means of in-class sessions with students. Here, students prepare their plan for the implementation of a single complex application that will be developed in stages during the quarter. Although the entire process is interactive (between the instructor and the students), we are guided by the planning approach found in [Sigwart, Van Meer, and Hansen, 1990:chap 4].

### Project Management in the First Course

The Applications Development Cluster in our IS curriculum consists of a three course sequence, terminating with data base applications. Although the first two courses presently use COBOL on an IBM mainframe, a focus on analysis and design permits migration to any suitable tool for implementation. Students entering the sequence have completed the Fundamental Concepts Cluster modeled for the ACM's Computer I and II. Students have also obtained elementary skills for use of a DBMS in a prerequisite end-user course in microcomputer tools (IS'90, Cluster A; IS'93,

Cluster B).

The integration of concepts of project management in the first course of the cluster is accomplished in five phases:

- definition of the project,
- formation of a benchmark for evaluation,
- completion of the first phase of the project,
- development of a class project plan, and
- periodic review and assessment of the project plan.

Project Definition. During the first week of the quarter, students are given the specifications for the final project. The choice of application, a multiple level control break problem with table look-up and final reporting is of sufficient complexity to explore the capabilities of the programming tool and to present associated knowledge elements.

#### Developing a Benchmark for Project Evaluation.

The first assignment for the quarter requires using a relational DBMS to develop procedures to analyze test data for the final project. Several merits for this approach are listed below.

- Because the DBMS requires essentially no programming in order to produce a control break application report, students have more confidence that these results are correct.
- At each stage of the application development, the results may be compared with those obtained from the DBMS.
- Since the procedures have also been implemented with a DBMS, changes in test data are easily captured by the DBMS.
- By comparing student program results with those from the DBMS



application, the visible emphasis on "testing" and "validation" confirm the importance of these concepts at each stage of development.

- Finally, the objectives of the project are accentuated.

The First Phase. During the second week of the class, students are given a non-credit programming assignment. This assignment, a simple read and write, gives the student an opportunity to focus on the overall structure of an application; this includes elementary data and file structures, basic control constructs, basic methods for input and output, and programming and design standards. A sample program, an in-class session on problem-solving techniques, and the application of structured methods to obtain a design help students complete this assignment. This assignment is evaluated for structure and for compliance with programming standards; the design is evaluated for proper use of symbols; and output is evaluated for completeness and correctness. The final score of this assignment is not recorded so the students have a chance to acquire these skills without incurring a penalty.

After completing this first phase, the students will

- be able to use the development tool (COBOL), structured design techniques, and programming standards and
- have completed a first step towards the final project.

#### Developing a Class Plan for the Project.

To determine the general structure of the application, the final product is analyzed according to the Michael Jackson methodology (data structure design) as described in [Bell, Morrey, and Pugh, 1987]. Students are instructed to examine the specifications of the final product in order to

discuss how to complete it by extending the first phase. This analysis identifies what needs to be done (*activities*), and the relationship among these components determines grouping of activities (*phases*). With faculty guidance, students also identify knowledge units (*resources*) from the phases; these include control break logic, table search, design techniques, and syntax requirements. With each phase there is an associated application product (*milestone*). Scheduling needed to complete each phase involves coordinating activities which include phases, lectures, and class examination.

At the conclusion of these in-class planning sessions, the students and instructor will have reached a general agreement on the number of phases, the output and testing requirements of each phase, and the scheduling of lectures needed to teach the students the concepts necessary to complete the phases.

The remainder of the course is event driven with the instructor playing multiple roles: lecturer, mentor, reviewer, client, manager, and facilitator.

Based on the outcome of the planning sessions, the instructor prepares the specifications for each phase product. Each specification contains a set of objectives for that milestone, a description of how this milestone is an extension of the previous one, the output requirements, and the way the DBMS output may be used to validate the output. The class and the instructor agree on a suitable time-frame for each phase product; they all understand that the final product has an unalterable deadline.

Periodic Review. A review is conducted at the conclusion of each phase to discuss any changes that may be needed in the original plan in order to complete the next phase. If

students fail to recognize an issue, the instructor can always initiate the topic by posing the question, "How would you handle ...?"

## **A Sample Plan**

A description of possible student-defined phases and influencing course objectives is outlined below; it begins with the second phase.

A Second Phase. The second phase typically extends the first by adding arithmetic computations and field editing. Presentations are given for techniques and syntax needed to complete these activities. The DBMS output is used to validate output data.

A Third Phase. A third phase involves the addition of one module for final accumulations and another for final reporting. Page control logic is inserted in the detail-processing module. Students identify knowledge requirements for implementation of counters, accumulations, and decision control syntax. With these topics as an objective, the instructor should expand the presentations to cover related issues, *e.g. determining the largest value of a field on the file.* The DBMS is again used to validate totals. Page control logic is verified by examining the output.

A Fourth Phase. Phases developed by each class begin to differ as the project continues. A fourth phase usually incorporates control break processing at this point. By using a nested procedure for control break logic, students extend the logic they have already been using to process the file. The technique is easily expanded to multiple levels of control breaks. By assigning the design of a variety of control break exercises [Shelly and Cashman, 1977] as homework, students receive reinforcement in control

break design and a variety of group summary and exception reporting techniques. Depending on the DBMS tool, two or more levels of grouping can be requested to help validate the output from this phase.

A Fifth Phase. The remaining phase(s) usually concentrates on table processing concepts such as loading and searching. The number of phases and their composition depend on the class. If the DBMS supports multiple table reporting, its output can be used for validation. Otherwise, additional instructions can be given. In any case, this assignment may be included as part of a first assignment in the database course.

## **Reinforcement in the Remaining Courses of the Cluster**

To adhere to the spiral learning principle of IS'90, the project management concepts are revisited in the remaining two courses in the cluster.

The Second Course. The focus of the second course is systems development. The instructor demonstrates to the class that the application developed in the first course is just one component of a system of applications. The class is divided into small programming teams of two to four students; each team is assigned a component of the system. In-class discussion and planning is confined to component integration. Project planning at the component level is a team assignment. Teams are required to submit planning documents which include:

- a list of activities to be completed,
- a set of phases for testing and review,
- individual assignment responsibilities, and
- deadline dates for the completion of the activities.

The instructor reviews the proposed plans and offers advice on their feasibility.

The Third Course. The development of more complex systems with a relational DBMS is the focus of the third course in the cluster. Teams must

- create their own problem statement,
- prepare a needs analysis,
- design a database solution,
- establish a plan for testing,
- implement and validate the solution,
- completely document the system, and
- make individual formal presentations.

The instructor provides a general table of contents for the final documentation as a guide for identifying high-level activities. The instructor reviews all major phases of a team's project.

## Conclusions

Summary. The Applications Development Cluster occurs between the Fundamental Concepts and Systems Development Clusters and is pivotal to the success of the curriculum. By using a teaching method that emphasizes managing the development of an application, students are able to reflect on the Fundamental Concepts Cluster from a different perspective (the spiral learning principle). By approaching the development of an application as a project requiring careful analysis and planning, students have an opportunity to gain practical experience in preparation for the Systems Development Cluster. The comprehensive formal presentation of project management concepts in Cluster E - Systems Development, builds on the common experiential background.

By introducing project management techniques in this manner, we not only comply with the requirements of the IS'90 curriculum, but also help the students to

understand a way to manage the development of an application. The approach described has been undertaken in three successive quarters with increasing success. These planning skills are reinforced in the second and third courses of the cluster when students must develop their own project plan.

Benefits. Some additional benefits of this teaching method are listed below.

- Students are not left to themselves to discover a method for managing the application development process.
- The Jackson method for program development is demonstrated in the analysis and design stages and reviewed for each phase.
- Top-down structured development and integration are demonstrated by the process. Stubbing is practiced beginning with the second phase.
- Testing and validation are accentuated by the use of a DBMS.
- Integration testing is tested at each phase.
- As phases become more complex, students are encouraged to practice what they have been taught. They apply the principles they have learned to develop their own intermediate phases.
- Once the class plan has been established, student dialogue enhances their understanding of the plan.
- Instead of beginning a new assignment with each phase, students add to their existing one; thus reducing development time.
- Similar applications may be introduced to reinforce analysis and design concepts.
- Since students are more involved in the planing, they are more interested in learning the necessary knowledge units. The students have a better

overview of the course and how its concepts are related.

- Students exhibit less difficulty in meeting the deadlines that they have set for themselves.

Lessons learned by the Instructor Although the students become more actively involved in planning their course of study, the instructor must be careful NOT to teach only the phases. A broader coverage of the material must be provided in order to address not only the current needs of the students, but also the requirements of the future. The stability of lecture material in the course is achieved while still allowing for many alternatives for future projects.

Because of time constraints, we had to compromise on the complexity of the project in the first course. The recommendation of a structured supervised laboratory as described in [Longenecker and Feinstein, 1991] has been approved for future implementation.

Final Notes The course text [Uckan, 1992] provides a software engineering approach to teaching COBOL. It is an excellent resource, providing complete and comprehensive coverage of syntax, design, data and file structures, and database concepts. All but a few of these sections are covered during the first two courses.

## REFERENCES

1. Bell, Doug, Ian Morrey, and John Pugh. *Software Engineering: A Programming Approach*. Prentice/Hall International, 1987.
2. Bergland, Glenn D., and Ronald D. Gordon. *Software Project Management. Software Design Strategies, Tutorial, Second Edition*, (1981), 3-8.
3. Bloom, Benjamin S., et al, *The Taxonomy of Educational Objectives: Classification of Educational Goals, Handbook 1: The Cognitive Domain*. McKay Press, New York, New York, 1956.
4. Daigle, R. J. *Teaching COBOL with Generic Design*, SIGCSE Bulletin, V17,3, p.12-16. 1985.
5. Feingold, Carl. *Fundamentals of COBOL Programming, Third Edition*. W. C. Brown. 1978.
6. Feingold, Carl and Louis Wolff. *Fundamentals of Structured COBOL Programming, Fourth Edition*. W. C. Brown. 1983.
7. Feingold, Carl and Louis Wolff. *Fundamentals of Structured COBOL Programming, Fifth Edition*. W. C. Brown. 1988.
8. Grauer, Robert T., and Marshal A. Crawford. *The COBOL Environment*. Prentice-Hall, Inc. 1979.
9. Grauer, Robert T. *COBOL, A Vehicle for Information Systems*. Prentice-Hall, Inc. 1981.
10. Grauer, Robert T. *Structured Methods Through COBOL*. Prentice-Hall, Inc. 1983.
11. Janossy, James. *COBOL: A Software Engineering Introduction*. The Dryden Press. 1989.
12. Longenecker, Herbert E. and David L. Feinstein, Editors 1991c. "IS'90: The DPMA Model Curriculum for Information Systems for 4 Year Undergraduates", published by Data Processing Management Association, Chicago, 1991.
13. Longenecker, Herbert E., et al. "IS'93: The DPMA 2 Year Model Curriculum - A Natural Evolution From IS'90, A Strong Information Systems Foundation", *Proceedings of the International Academy for Information Management*, (1992), 253-260..

14. McCracken, Daniel D. *A Guide to COBOL Programming*. John Wiley & Sons. 1963.
15. McCracken, Daniel D. *A Guide to COBOL Programming, Second Edition*. John Wiley & Sons. 1970.
16. Philippakis, Andreas S. *Information Systems Through COBOL*. McGraw-Hill. 1974.
17. Philippakis, Andreas S. *Information Systems Through COBOL, Second Edition*. McGraw-Hill. 1978.
18. Shelly, Gary B. and Thomas J. Cashman. *Introduction to Computer Programming Structured COBOL*. 1977.
19. Shelly, Gary B. and Thomas J. Cashman. *Advanced Structured COBOL Program Design and File Processing*. 1978.
20. Sigwart, Charles D., Gretchen L. Van Meer, and John C. Hansen. *Software Engineering, A Project Oriented Approach*. Franklin, Beedle, & Associates, Inc. 1990.
21. Welburn, Tyler. *Structured COBOL: Fundamentals and Style, First Edition*. Mayfield Pub. Co. 1981.
22. Welburn, Tyler. *Advanced Structured COBOL: Batch, On-Line, and Data-Base Concepts*. Mayfield Pub. Co. 1983.
23. Whitten, Neal. *Managing Software Development Projects: Formula for Success*. John Wiley & Sons, Inc, 1990.
24. Uckan, Yuksel. *Application Programming and File Processing in COBOL, Concepts, Techniques, and Applications*. D. C. Heath and Company. 1992.

# **APPLICATIONS DEVELOPMENT IN CLIENT-SERVER ENVIRONMENT: A STEP IN THE RIGHT DIRECTION FOR CIS CURRICULUM**

Mayur R. Mehta, Southwest Texas State University, San Marcos, Tx. 78666  
George W. Morgan, Southwest Texas State University, San Marcos, Tx. 78666

## **ABSTRACT**

The primary thrust of educational programs in Computer Information Systems (CIS) is to prepare graduates to analyze, design, implement and maintain information systems in technologically dynamic business environments. While it has been in existence for years, CIS is still considered an emerging academic discipline with goals, subject matter, and problem solving processes sufficiently different from other computer-related disciplines to warrant special attention. Continual updates to the curriculum are warranted due to rapidly changing business environments, made even more complex by rapidly changing computer technology whose use is proliferating throughout the business community. As Information Systems educators, we are challenged to predict what future occupational skills will be needed by our graduates, and to refine our curriculum accordingly. This paper describes the efforts of the department of CIS at a four-year, medium-sized university to update its CIS curriculum to be more in-line with the demands of the work place. Details of a major product this effort culminated into are also presented.

## **INTRODUCTION**

Increased reliance of businesses on modern technology, particularly personal computers, alters the job opportunities our graduates are likely to encounter. Probably nothing has had a greater impact on the way businesses manage their information than the advancement in the microcomputer technology. Downsizing applications from large mainframes to PC/Workstation environments is a current trend that is being discussed daily in the IS media. The primary force driving this downsizing effort is the maturing of the many client-server tools, such as DBMS, GUIs, and application toolkits to support networks and event-driven programming. Such trends presents CIS educators new opportunities to refine and update the curriculum content. Keeping in touch with the work place as a way to identify these

opportunities cannot be over emphasized . As the work place evolves, so must our programs. We must increasingly keep in touch with the work place to know what tools, skills and knowledge our graduates actually use. One way to accomplish this is through Industry Advisory Councils. This paper outlines our experiences in our association with an advisory council and the benefits derived from that working relationship.

A one-day CIS Advisory Council meeting was held on campus, bringing together the representatives of industry and the CIS faculty. The marathon workshop, among other things, established consensus on two major points: (1) The primary goal of the department was to prepare entry-level applications programmer/analysts for the modern commercial environment, and (2) that there were apparent gaps in the our

current curriculum's ability to impart the knowledge and skills needed to make our graduates more marketable and to help them advance in their careers. The major flaw in the current curriculum was found to be a lack of exposure to applications development in downsized environments, a trend of the 90's. There was a general consensus among the participants with respect to this technological shift in the work place and a definite need to align the curriculum with this shift. With the help of the advisory council, we were able to establish broad guidelines to direct our curriculum refinement effort. The workshop concluded with the formation of a faculty group to develop specific plans for identifying and making the necessary refinements to the curriculum.

### **Statement of Problem**

The challenge was to refine the curriculum content in such a way that it would emphasize concepts and methods of developing business information processing applications in a variety of computing environments and, if possible, without increased academic requirements for students. The proposed curriculum refinement should include use of computer-based development tools for designing and implementing specific business applications in client-server, microcomputer, workstation, multi-user and multi-tasking environments. In addition, it should allow students to extend their knowledge of systems analysis and design by undertaking some type of close-to-the-real-world computer case study.

### **Process**

The faculty approached the problem(s) from an end-user perspective. Based on the broad guidelines established with the

help of the advisory council, a detailed list of skills perceived to be important for our graduates to possess prior to entering the work force was developed. The skills list was further grouped into knowledge clusters. The existing curriculum was evaluated against this set of knowledge clusters to determine where improvements needed to be made to align our curriculum with the work place requirements. The individual faculty members were asked to formulate specific course curriculum that in their opinion provided the best coverage of the knowledge clusters without significantly increasing students' academic requirements.

Eight alternative plans were formulated. These ranged from a curriculum emphasizing mainframe-based programming to a curriculum emphasizing pc-based applications development; and from very structured (few electives) to fairly unstructured (more electives). While the proposed alternatives differed in many aspects, they were all in agreement with one major refinement. They all agreed upon a definite need for an advanced course that emphasized applications development in downsized or client-server environments, integrating knowledge clusters pertaining to systems as well as database analysis, design, and implementation.

After several discussion sessions, the faculty were in strong agreement that a modification to the current CIS curriculum should be made. Three major factors emerged as leading the list of reasons.

First, the recent trend in technology has been towards smaller, yet more powerful computer systems; moving information processing out of the computer room and placed in the hands of non-technical end-

users. As a result, many companies are downsizing their information processing applications where only large databases reside on mainframe computers (acting as database servers) while most information processing tasks are directly conducted by end-users working on microcomputers or workstations (acting as clients) that are networked with mainframes. In addition, to support end-user computing, user-interfaces have changed dramatically from the old character-based, command-driven interfaces to icon-driven GUIs. Such technological trends have dramatically altered the way systems are designed and implemented today. Consequently, modern information systems have become more complex and require increased consideration for the interaction among users, hardware, and software. Newer design concepts, tools, and methodology have evolved to support such technological shifts. Familiarity with these is a must for CIS graduates to succeed in their chosen career.

Second, the continuing advances in the field of microcomputer technology has placed the power of information processing in the hands of end-users and systems analysts. Consequently, complete business processing applications can now be developed to work within downsized computing environments. These advances have also resulted in a greater availability of user-oriented computer tools to support quick development of business information processing applications for implementation on microcomputer/workstations systems within a multitude of computing environments, ranging from single-user/single-tasking to multi-user/multi-tasking environments, and from standalone to client-server computing environments.

Systems analysts of today, thus, have a

much bigger arsenal of sophisticated tools and techniques at their disposal to support their applications development effort.

Third, most CIS graduates will enter the job market to work as programmer/systems analysts where they will be required to analyze, design and implement systems to support information needs of a variety of users working in a variety of computing environments. Thus, familiarity with concepts, methods and tools for developing business information processing applications in such environments will play an increasingly important role in how they perform on the job. Knowledge of such concepts, methods, and tools in the downsized computing domain will provide them with a competitive edge.

#### **Outcome of the Curriculum Refinement Effort**

A one-semester, senior-level, course on Advanced Microcomputer Systems (or more precisely a course stressing Applications Development in a Client-Server Environment) was added to the curriculum. In addition, an effort to restructure and resequence existing courses was undertaken. The end-result of our curriculum refinement effort was to focus on the technological shift towards client-server as well applications development in a multi-user, multi-tasking, pc-based and GUI-dominated environments. The topical outline of the proposed course and proposed changes in the curriculum are presented as Attachments 1 and 2, respectively.

#### **REFERENCES**

References available upon request.



## **ATTACHMENT 1**

### **TOPICAL OUTLINE OF THE ADVANCED MICROCOMPUTER COURSE** (Applications Development in a Client-Server Environment)

- 1. COURSE OVERVIEW**  
(Time Devoted = 10% of the semester)
  - 1.1 Changing Role of Technology
  - 1.2 Move towards Downsizing, Multi-Tasking Windowing and Networked Computing Environment.
  - 1.3 Evolving Concepts, Tools, and Methodology.
  
- 2. INTRODUCTION TO MICROCOMPUTER/WORKSTATION TOOLS:**  
(Time Devoted = 15% of the Semester)
  - 2.1 Introduction to Operating Environments - DOS, OS/2, Windows
  - 2.2 Introduction to Windows
  - 2.3 Applications Development under Windowing Environment
    - 2.3.1 Introduction to MICRO-STEP
    - 2.3.2 Introduction to code-generators
    - 2.3.3 Introduction to multi-tasking
    - 2.3.4 Integrated Application Development using standard productivity tools - Spreadsheets (MS-EXCEL or LOTUS-123), DBMS's (Paradox or dBASE), Graphics, & Programming Languages
  
- 3. APPLICATIONS DEVELOPMENT UNDER NETWORKED ENVIRONMENT.**  
(Time Devoted = 45% of the Semester)
  - 3.1 Introduction to client-server systems
  - 3.2 Applications that share data/information
    - 3.2.1 Front-Development Tools  
VisualWorks, PowerBuilder, PowerMaker, Enfin/2, SQL\*Windows
  - 3.3 Applications that share programs
  - 3.4 Applications for remote sharing of data, programs, and resources.
  - 3.5 Applications to query remote databases for local information processing using SQL and RDB.
    - 3.5.1 Back-End Development Tools  
Oracle version 7.0, Sybase, SQLBase, Quest
  
- 4. ADVANCED MICROCOMPUTER SYSTEMS DEVELOPMENT PROJECT:**  
(Time Devoted = 30% of the Semester)
  - 4.1 During the last one-third of the semester, students will undertake a team-oriented computer case study requiring student teams to analyze, design and implement specific business applications using the application development tools for "downsized" environments. Requirement is to develop applications that incorporate all the concepts, tools, and methodology covered in the course.

## ATTACHMENT 2

### Proposed New CIS Curriculum

<b>REQUIRED COURSES (21 HOURS)</b>		<b>PREREQ.</b>
CIS 2324	Techniques of Computer Programming	NONE
CIS 2371	Introduction to COBOL Programming	2324
CIS 3372	Data Management & Retrieval	2371
CIS 3374	Systems Analysis	2371
CIS 3375	Advanced COBOL Programming	3372
<b>CIS 4322</b>	<b>Computer Systems Development &amp; Design</b>	<b>3374, 3375</b>
	<b>OR</b>	
<b>CIS 4323</b>	<b>Advanced Microcomputer Systems</b>	<b>1323, 3374</b>
CIS 4382	Computer Database Systems	3372
<b>ADVANCED ELECTIVES (6 HOURS)</b>		
CIS 3322	Programming Tech. & Machine Org. (BAL)	2324
CIS 4344	Information Resource Management	1323, 3370, & MGT. 3303
CIS 4346	Decision Support Systems	1323, 3374
CIS 4348	Business Data Communications Processes	1323, 3374
CIS 4349	Fourth Generation Languages	2371, 3370
<b>CIS 4322</b>	<b>Whichever one is not taken as a required</b>	
<b>OR</b>	<b>design and development course.</b>	
<b>CIS4323</b>		

# Better End Users--IS Design Issues for the MIS Class

Kathleen Moffitt  
Information Systems/Decision Sciences  
California State University--Fresno  
5245 N. Backer, Fresno CA 93740-0007  
(209) 278-2415

## Abstract

Classroom training in software traditionally concentrates on instruction in keystroke and command-level use of software and has neglected other areas critically in need of attention, namely design and control. An approach to educating undergraduate and MBA non-information system students in the importance of spreadsheet design and control methods is presented.

## Introduction

Aside from word processors, spreadsheets are the most widely used software package in the business environment. Most recent graduates of business schools have had some exposure to spreadsheets; others have learned to use spreadsheets through self-training or organizationally provided employee training programs. This training, whether formally or informally undertaken, generally concentrates on learning the use of keystrokes/commands and features available within a specific piece of spreadsheet software. In spite of this sometimes considerable training, end user developed spreadsheets continue to have a dangerously high rate of errors (Brown & Gould, 1987; Gilman & Bulkeley, 1986; Gingrich, 1990; Hayen & Peters, 1989; Whitehurst & Roberts, 1990). The severity of this problem will only grow as reliance on end user application development increases (Lu, Litecky & Lu, 1991).

While knowledge of how to use spreadsheet software commands and features is a necessary condition for reducing the rate of errors, it clearly is not a sufficient condition. A new approach encompassing both command and feature-based training as seen in, for example, Kleim, 1991; Nelson & Cheney, 1987; and Sein, Bostrom & Olfman, 1987; and design as seen in, for example, Edge & Wilson, 1990; Guimaraes & Ramanujam, 1986; Kee & Mason, 1988; Mann, 1990; Pearson, 1988; Ronen, Palley & Lucas,

1989; and Stone & Black, 1989, is needed. Still others, such as Gass, 1990; Hayen, Cook & Jecker, 1990; and Weida, 1990, identify a need for a learning environment that addresses the more conceptual issues of modeling business problems and decisions.

Understandably the first, training in the use of software package commands and features, is the easiest to address and accomplish. The second, training in design, is more difficult to accomplish, as individual preferences, organizational procedural requirements, and the nature of the problem environment, vary significantly. Nevertheless, there are numerous design concepts that are appropriate to almost all environments. The last, the modeling of business problems and decisions, is essentially beyond a mere MIS class, whether undergraduate or graduate, and is more appropriate for a business modelling/decision support systems class. Unfortunately, you rarely see a non-information systems major in such a class. This shouldn't be problematic since our entire undergraduate and MBA education should be producing graduates who are capable of conceptually modeling business problems and decisions (in non-computer terms). In reality, however, it is problematic of most business school education, especially at the undergraduate level.

Four conditions are identified which must exist in order for a spreadsheet to be appropriately and correctly used as a problem solving tool. The

student/end user needs to have the ability to: 1) identify a business problem/decision which can be aided through technology (in this case a spreadsheet); 2) correctly translate the problem/decision into a form where use of a computerized spreadsheet is appropriate; 3) correctly use the spreadsheet as a tool (i.e., correct use of commands and features; appropriate use of testing, debugging and error detection techniques; and a design and documentation appropriate for its intended or potential use); and 4) correctly interpret the output. Our educational process should address all four. Conditions three and four, and to a lesser extent condition two, are addressed with the approach presented here.

As information systems educators we look at the four conditions and think to ourselves--there's nothing new here. I lecture on all of these issues in my MIS class and students learn how to use a spreadsheet in the introductory computer class. Our students know how to do these things. The reality is that we are fooling ourselves into complacency by thinking along these lines. As mentioned earlier, end user developed spreadsheets have dangerously high error rates, ranging from forty to one-hundred percent. Similar error rates are found in MIS classes where the proposed approach has been used. MBA management information systems classes have an error rate range of forty-two to seventy-two percent and undergraduate management information systems classes have error rates ranging from sixty-six to eight-three percent. Errors are defined as any calculation mistake that effected the output of the spreadsheet by a consequential amount.

The abilities and skills identified in the four conditions cannot be taught through lecture or reading. They require having students struggle through hands-on exercises, having the instructor use the exercises as lecture material after, not before, the struggle, and having the instructor encourage significant classroom discussion over the merits of specific design considerations and how and why students did what they did.

A specific approach for teaching spreadsheet design concepts follows. In addition to teaching design, this approach forces students to tune up their rusty spreadsheet skills, learn new features of spreadsheets, and learn new ways of doing familiar things. It also provides some conceptual/modeling experience for students, but this is not the overall purpose of the approach.

### **The Approach**

The approach involves giving students a multi-week spreadsheet case assignment. The assignment is broken down into two parts. Students hand in the first part of the assignment at a specified time, the instructor grades and comments on the work, hands it back one week later, and lectures at length about the problems seen in the assignment and what can be done to avoid these problems in the future. It is important for there to be significant class discussion at this time. It provides for sharing of ideas and experiences and helps anchor the learning. The student takes the returned assignment (part one), reworks it according to good design principles, does some additional analysis with it, and hands it in as part two of the assignment. The new analysis is designed to reinforce the importance of good design by being easy to accomplish with good design and time consuming and error-prone without good design. Included with part two is the requirement for the student to critique the good and bad aspects of the part one design and discuss what they have learned about spreadsheet design. The overall grade is split between the two assignments. This allows students who do poorly on the first part to get a good grade on the second part, and avoid overall discouragement.

The case clearly states that the spreadsheet will be used by another person (a manager) and specific values will need to be changed to study their impact on the decision which will be made. In spite of these hints, students typically put little to no thought into constructing the spreadsheet for usability by another person or reuse by themselves. They approach it squarely as a

computational problem, not as a design problem. I have seen the same during a recent academic leave spent working in industry.

The case which is used does not need to be highly complex, and actually works best when it is of moderate complexity. Since the actual purpose of the assignment is to demonstrate the high rate of spreadsheet errors and good spreadsheet design, not an understanding of a computationally complex business problem, an excess of complexity is counter-productive. In fact, students are typically quite surprised to lose points when all of their computations are correct, but design elements are poor.

It is important for the chosen or developed case to have formula development and information presentation/gathering requirements where computational, cognitive or design errors are likely to occur. This allows the richness of the case to be in its content not in its complexity. The following provides a list of suggestions for concepts to include in a spreadsheet case/exercise. A brief discussion accompanies each suggestion.

1. Mix the units of measure for data provided in the case. Present some data in thousands, other data in units; present data as weekly, monthly, quarterly and yearly.

- This requires the student to be cognizant of the data's unit of measure.
- Serious computational errors can result from a failure to recognize these distinctions and adjust for them in formulas.
- Serious modification errors can occur if formulas which adjust data to compatible units are not properly documented.
- Serious information display and interpretation errors occur when the student fails to label the spreadsheet with the correct unit of measure.

2. Provide values that are used throughout the spreadsheet.

- Demonstrate the usefulness of a data input area and the value of absolute cell referencing for values that are used throughout the spreadsheet.
- From an overall design, error-prevention and reusability standpoint, failure to use cell referencing is the most serious design error seen in spreadsheets.
- The second part of the spreadsheet assignment should require running at least two sets of different values through the spreadsheet. This clearly drives home the usefulness of data input areas and the automatic recalculation which takes place with cell referencing. As one student told the class, if you use an input area, it takes longer to print the spreadsheet than to make the changes.

3. Have some formulas that require knowledge of the hierarchy of arithmetic operations.

- This is critical as a formula can look correct in the sense that it has all the right numbers or cell addresses in it, and yet produce completely erroneous results if the user doesn't understand that multiplication occurs before addition, etc., unless parentheses are used.

4. Use percentages to see that they are entered and used properly

- Percentages must sum to 1.
- See if concepts behind conditional and joint probabilities are understood.

5. Have columns/rows of numbers that can be meaningfully summed and others that cannot be meaningfully summed.

- Anything that can be meaningfully summed should be, because the user of a spreadsheet should never have to get out a calculator to do something the spreadsheet program can do easier, quicker, and with less chance of error.
- Include one or more columns/rows that accumulate (have running totals) to

demonstrate that some columns cannot be summed because the results are highly inaccurate and misleading.

6. Have data needed for the assignment spread throughout the text of the case.

- Errors of omission are very common. By spreading out data, the user is encouraged to assess whether everything that is needed in the analysis has actually been included.

7. Require listing of formulas for key parts of the spreadsheet. It is shocking how many people calculate everything on paper and just put the values into the spreadsheet.

- Watch for rows/columns that don't add properly. This is a dead giveaway.

8. Pay special attention to information presentation as it can lead to serious interpretation errors or lack of usability.

- Decimal place formatting should be consistent within columns/rows of like values/attributes.
- Decimal place formatting should be used to send a message to the user. For instance, two decimal places visually leads the user to think in units, while one or three decimal places leads the user to think in thousands. This can help prevent misinterpretation of otherwise correct data.
- Information presentation can be used to lead the user logically through the spreadsheet—from input, through processing and finally to output. Data should be presented from the point of view of the ultimate user and not from that of the builder.
- Data presentation should clarify, not confuse. The spreadsheet, as well as all data groupings, columns and rows should be clearly labeled so that the spreadsheet will be understandable one day, one week, one month and one year from the time it was built.
- Don't ever assume that you will be the sole user of your spreadsheet and its output. Someone will always want to

use it for the same purpose or a slightly different purpose. Give them a product that is reusable, modifiable, and easy to use and understand.

9. Use simple error detection methods wherever possible.

- Provide examples specific to your case but include things such as row and column totalling and cross checks.
- Use reasonableness assessment of output. If the output looks strange for the data inputs, check your calculations. A good understanding of the data and the problem can go a long way toward helping to detect errors.

10. Require a transmittal memo from the builder (student) to the individual who will be using the spreadsheet output. This should require brief written interpretation of the output.

- Perfectly correct spreadsheets (from a calculation point-of-view) can be misinterpreted because of poor or inadequate information presentation or documentation, or because the user doesn't understand what has been done.

11. Require the student to assess, in writing, the good and bad points of their spreadsheet part one design and discuss what they learned about spreadsheet design. This is handed in with part two of the assignment.

- This helps reinforce the points of the exercise and provides the instructor with valuable insight into the assignment and the educational process.

This is not an exhaustive list of concepts to address in this type of assignment. It is simply what I have developed over the three years I have been using this approach to teaching spreadsheet design concepts to non-information systems students. Your approach will develop over time due to the problems you see, the nature of the exercise you develop and the information you receive from students during class discussion and from their written comments on what they have learned.

### **Strengths and Weaknesses of the Approach**

The greatest strength of the exercise is that students find it extremely useful and valuable. They realize they have learned a great deal because of the hands-on experience and that they could not have learned this from reading or lecture. Students who have weak spreadsheet skills struggle the most with the assignment but learn a great deal during a short period of time. Those who are frequent spreadsheet users often get jolted out of their state of overconfidence. The serious errors in this assignment are not confined to the less skilled spreadsheet users. Many "spreadsheet jocks and jockettes" have made serious and embarrassing calculation, interpretive or omission errors. Most students say they are completely rethinking how they use spreadsheets at work; others say they are going to start using spreadsheets because they see their usefulness for many tasks they have done previously by hand or that can be done better or quicker with a spreadsheet.

The greatest weakness of the approach is that you cannot just turn the students loose with it. You need to provide some structure for data groupings and output so that the assignment is gradeable and relatively consistent from a presentation point of view. If you don't do this, everyone will do their own thing, and it will detract from your ability to grade the assignment; to comment on specific points of value, detect errors, and make design suggestions; and will certainly affect your willingness to ever do the exercise again.

Another weakness is that a student's best resource while in school is his/her fellow students. Unfortunately, the exercise does get out so you need to continually modify/update it so that it remains a learning experience rather than a rehash of someone else's work from the previous semester.

Finally, neither a weakness nor a strength—it is important to adjust the assignment to the level of student knowledge. Some undergraduates can be nearly overwhelmed if too much is thrown at

them at one time. Several other faculty have adopted this approach, and are pleased with the results, but some have broken the undergraduate assignment into many parts done over an extended period. They do this to help students learn more about using the commands and features of the spreadsheet. Undergraduate software use skills are rarely as well developed as we would like to think. I have found the same to be true in industry.

### **Conclusion**

Current spreadsheet software training efforts are inadequate not because of what they cover but because of what they don't cover. In a business environment where most workers have access to or use technology as a business tool, we need driver's training that goes beyond "here's the steering wheel, there's the clutch and there's the brake". Current training implies to people that all they need to know or learn is how to issue a set of commands in a software package. The importance of this is not denied; it is analogous to learning vocabulary when a new language is being learned. But our language training doesn't stop at vocabulary and our software training almost always does. Our language training proceeds to context and structure and this is how meaning is conveyed and what allows communication to take place. We need to develop a similar approach to educating students and training employees in the area of end user computing. The price of not doing this may be difficult to measure but is nevertheless present, high and ever increasing.

### **References Cited**

- Brown, P. S. & Gould, J. D. (1987) An experimental study of people creating spreadsheets. *ACM Transactions on Office Information Systems*, 5(3):258-272.
- Edge, W. R. & Wilson, E. J. G. (1990) Avoiding the hazards of microcomputer spreadsheets. *Internal Auditor*, 47(April): 35-39.

- Gass, S. I. (1990) Model world: Danger, beware the user as modeler. *Interfaces*, 20(3):60-64.
- Gilman H. & Bulkeley, W. M. (1986) Can software firms be held responsible when a program makes a costly error? *Wall Street Journal*, August 4, 1986.
- Gingrich, G. (1990) User interface design: Need and methods for the comparative study of Lotus and IFPS. *Proceedings, Third Symposium on Human Factors in Information Systems*, J. Carey (Ed.), pp. 8-21.
- Guimaraes, T. & Ramanujam, V. (1986) Personal computing trends and problems: An empirical study. *MIS Quarterly*, 10(2):179-187.
- Hayen, R. L., Cook, W. F., & Jecker, G. H. (1990) End user training in office automation: Matching expectations. *Journal of Systems Management*, 41(3):7-12.
- Hayen, R. L. & Peters, R. M. (1989) How to ensure spreadsheet integrity. *Management Accounting*, LXX(10):30-33.
- Kee, R.C. & Mason Jr., J.O. (1988) Preventing errors in spreadsheets, *Internal Auditor*, 45(February):42-47.
- Kleim, R. L. (1991) What happens after software training? *The Office*, 113(2): 16ff.
- Lu, M., Litecky, C. R., & Lu, D. H. (1991) Application controls for spreadsheet development. *Journal of Microcomputer Systems Management*, 3(1):12-22.
- Mann, R. O. (1990) Designing spreadsheets that make sense. *PC Today*, 4(10):22ff.
- Nelson, R. Ryan & Cheney, P. H. (1987) Training end users: An exploratory study. *MIS Quarterly*, 11(4):547-559.
- Pearson, R. (1988) Lies, damned lies, and spreadsheets. *Byte*, 13(13):299-304.
- Ronen, B., Palley, M. A. & Lucas, Jr., H. C. (1989) Spreadsheet analysis and design. *Communications of the ACM*, 32(1):84-93.
- Sein, M. K., Bostrom, R. P. & Olfman, L. (1987) Training end users to compute: Cognitive, motivational and social issues. *INFOR*, 25(3):236-255.
- Stone, D. N. & Black, R. L. (1989) Building structured spreadsheets. *Journal of Accountancy*, 168(4):131-142.
- Weida, N. (1990) Training implications of spreadsheet modeling. *Managing Information Resources in the 1990s*, M. Khosrowpour (Ed.), Idea Group Publishing, Harrisburg, PA, p.147.
- Whitehurst, J. S. & Roberts, A. H. (1990) Problems associated with technically unskilled microcomputer users: Report on a survey of IS managers. *Managing Information Resources in the 1990s*, M. Khosrowpour (Ed.), Idea Group Publishing, Harrisburg, PA, pp. 108-113.



PARTNERING IS WITH THE BUSINESS COMMUNITY:  
THE INFORMATION SYSTEMS RESEARCH INSTITUTE

Dr. Jean B. Gasen  
Mr. Edwin E. Blanks

Abstract

Virginia Commonwealth University maintains a strong commitment to the surrounding business and community environment. As Virginia's largest major urban university, the university holds a unique position. From the Board of Visitors down to the individual departments, the university emphasizes that it is in a mutually beneficial partnership with the community in which it lives. This paper describes how the Information Systems (IS) partnership with the business community is a natural outgrowth of relationships already formed on higher levels of university organization and how the goals of the IS Research Institute fit well within the overall university mission. Several innovative programs are highlighted to demonstrate how business-academic partnerships can be mutually beneficial to both entities.

# **EDUCATING THE BUSINESS EDUCATOR: FACULTY INTERNSHIPS**

Thomas P. Loughman and Robert A. Fleck, Jr.

Abbott Turner School of Business  
Columbus College  
Columbus, Georgia 31907-2079  
(706) 568-2284

## **Abstract**

This paper describes several internships undertaken by the authors. As college business faculty they were invited to become involved in internships with local business organizations over several summers. Based upon their experiences and research, the authors present (1) the potential strengths and limitations of faculty internships, (2) benefits and drawbacks of internships to interns and sponsoring firms, and (3) educational implications of faculty internships.

## **INTRODUCTION**

Student internships have proven to be of value to students and to those firms which support internship programs. Students apply classroom learning to actual business situations, and firms have a low-risk opportunity to evaluate potential employees while indirectly supporting a local school of business. Faculty internships share some of the features of student internships but also display several differences. This paper explores the benefits and risks of faculty internships through the experiences of two faculty members who accepted summer internships. Recommendations are supplied for those who may wish to develop or participate in an internship program, with emphasis placed upon the teaching implications.

## **EXPERIENTIAL LEARNING**

Experiential learning has long been favored by many college professors. It combines classroom experiences with exercises designed to simulate the business environment under controlled conditions. Student internships are one method cited in the literature (Spinks, 1989) as a suitable vehicle for experiential learning. Student interns exposed to selected "live" events are expected to use frameworks developed in the classroom (text and lecture) to make decisions. Based on the decision outcomes, students and the sponsoring firm can assess the intern's level of preparation. Experiential learning benefits also accrue to faculty placed in internships. The faculty intern improves his or her ability to present text and lecture materials based on "live" events.

## **FACULTY INTERNSHIPS**

One important element of the partnership between our institution and the local business community is faculty internships. Faculty internships provide opportunities for faculty to work with a local company and renew or improve applied business skills. For faculty with limited exposure to daily business practices, internships provide opportunities to test academic skills. For other faculty, internships may test cherished assumptions and help establish new directions for research. The sponsoring businesses also benefit from the exposure to new frameworks of thought, and students can benefit from the learning and revitalization experiences of their faculty.

## **INTERNSHIP STRUCTURE**

A key element in the internship program is the Executive Conference. The Conference consists of fifty top executives and business leaders from the local community who are committed to excellence in business education. Internships are arranged through the offices of the dean and the faculty member's department chair in consultation with the Executive Conference. First, faculty skills and desires are assessed and matched with professional needs in participating firms. The dean then makes the initial contact with the firm, after which the faculty intern is responsible for all future contacts.

When a firm agrees to accept a faculty intern, the intern and firm confer on how best to use the faculty member's talents and how to maximize the faculty member's experience with the company. On occasion, the intern will submit a proposal to company management, after which further discussions

may be held to insure that both the company and the intern understand what is to be done during the internship.

Faculty interns work for two months and receive the equivalent of their two-months' summer teaching salary. These funds are paid directly to the intern, who is considered an employee of the firm, not the college, for the internship period. Any further consulting agreements are negotiated between the company and the intern.

The following three internships describe the activities and responsibilities experienced by one of the authors. The other author also participated in the internship program and experienced similar events. Conversations with other faculty interns indicate that the experience described here are fairly common. These descriptions can guide other faculty members considering internships.

## **INTERNSHIPS**

### **Internship 1**

This internship took place in a large, diversified manufacturing company. Because of the intern's background in communication, the primary tasks were writing articles and reviewing written marketing materials. Additional duties involved presenting seminars on writing and speaking skills and conducting meetings.

This internship was a first for both parties. While some contact took place after the initial negotiations and before the first day of "work," both the intern and the company were uncertain about expectations and tasks. The first month was spent on short-term tasks while long-term projects were discussed. Work continued on some of the

projects for over a year after the internship officially ended.

### **Internship 2**

The second internship was with a full-service communications company. Because of the experience gained in the first internship, establishing tasks was much easier and took less time to accomplish. Duties involved measuring communication flow, developing procedures manuals and orientation materials, and reviewing company publications such as annual reports. Because this internship was better planned than the first, most of the projects were finished within the designated internship period.

### **Internship 3**

Internship 3 was with a manufacturing firm with international markets. Based on experiences of the first two internships, the faculty intern sent a formal proposal to the firm. The proposal clearly outlined tasks to be performed by the intern and the benefits to be derived by the firm. It also clarified limitations so that expectations were reasonable and achievable.

### **Internship Summary**

The internships described passed through stages. The first stage exhibited amorphous boundaries and uncertainties by both the intern and the firm. The desire for more immediate productivity led to better planning for the second internship experience, improving the definition of tasks to be accomplished. The third internship built upon the experiences of the first two and placed more responsibility for task management with the faculty intern. Management of the sponsor company was thus in a better position to provide positive

guidance to the intern, and the intern was able to demonstrate how his expertise and knowledge would contribute to the operation of the firm. The internships thus exhibited a movement toward a more clearly defined structure with the intern taking increasing responsibility for determining the tasks and outcomes. Detailed explanations of the internship processes are given below.

## **INTERNSHIP BENEFITS**

### **Benefits to Business**

The firm, while taking a financial risk, has its risk constrained by time. For the expenditure of funds, the firm has a full-time internal consultant. The intern can help relieve the company of tasks for which there are insufficient professional personnel available. (Cheatham, 1986)

The faculty intern brings special skills to the internship that usually are not available within the company. While these skills may be based on theory, they do provide the framework for applications. The theoretical, critical approach to problem solving often leads the intern to ask questions that would otherwise not be thought of or asked by other members of the organization. Management is thus provided with insights into operations that may be more objective than those provided by normal internal channels. The intern must therefore be careful not to be the bearer of messages constructed by others for political gain.

The sponsoring business also indirectly enhances the academic preparation of the pool of college graduates from which it hires its employees. It does this by allowing the faculty intern to develop pertinent experiences to support classroom

presentations. This point is especially important to regional urban institutions where the majority of business graduates will be employed by local firms.

### **Benefits to Faculty**

The primary benefit to faculty is the experience which can translate into long-term consulting relationships, research projects, and improved classroom presentations. (See Krogstad, 1981, for the effects of faculty residencies on improved teaching evaluations.) These classroom presentations can be supported through case examples of local firms. For instance, during one internship the faculty intern was asked to rewrite product assembly instructions. When he discovered a faulty gasket among the assembly materials and tried to call the 800 "hot line," he was informed that the number had been disconnected, with no other information given.

While this experience provides a cogent example of internal miscommunication in a large firm, it must be handled with sensitivity when presented to a class. (See Lee, 1988, for additional comments on confidentiality issues in faculty internships.) Case examples based on internships must be approached differently from cases obtained from publishing houses. While students can identify with local firms, the disadvantage is they may be associated with the firm through employment or family ties. Association presents two problems: 1) students may not be able to analyze objectively the specific corporate problem under discussion, and 2) classroom discussions may filter back to the firm.

Internships can also provide excellent opportunities to develop contacts in the business community. While funding rates

for internships are usually lower than consulting rates, if summer teaching opportunities are minimal, internships provide alternative income.

As academic sabbaticals are designed to refresh one's perspective by either working on research or teaching at another institution, internships provide much the same relief, mostly because of the changed work patterns and intellectual requirements. Also, not having a stack of papers to grade is surely an intangible asset of the internship.

### **Benefits to Students**

Students benefit indirectly but significantly from faculty internships. Faculty are clearly more enthusiastic about their academic disciplines after internships and can reinforce concepts based on current experiences. The business contacts made by faculty interns also serve as a ready pool for guest speakers in classes and potential employment contacts for students upon graduation. Faculty also can call upon their internship experiences when providing career counseling. (Krogstad, 1981)

## **POTENTIAL DRAWBACKS**

Internships are not without possible penalty to the firm, academic institution or faculty member. The firm is at risk financially for the intern's salary. In addition, a faculty member with poor inter-personal skills can damage relations built on personal trust.

The academic institution which recommended the intern desires a positive and continuing relationship with the firm. A failed internship may cause problems for student placement and future financial

support of the school by the firm. Less-than-tactful classroom presentations of problems within local firms can further erode corporate trust of and support for the academic institution.

The faculty member is at greatest risk. Poor performance will minimize future opportunities for consulting and may have repercussions affecting his or her reputation in the academic community. Also, the time spent on the internship is usually time away from colleagues and academic pursuits, the consequences of which can be significant to junior faculty, especially with regard to questions of tenure and promotion. Other faculty will be concerned about impacts on merit pay and retirement. Since the intern is paid by a company, internship pay does not contribute to the retirement salary base.

Another problem experienced by the authors was adjusting to different working rhythms. Academics typically exercise a great deal of control over their own schedules. They may work in short intense bursts, as when preparing for and presenting course materials, or they may work uninterrupted for long periods. However, when an internship is set up to accommodate the time constraints of the sponsoring company, the intern may have only minimal control over time management. He or she may begin to feel the company is paying for time spent sitting at a desk, not the intern's productivity. Interns may have difficulty developing safeguards against interruptions, especially since the internship itself may be something new in the firm and the employees may want to "pick the intern's brain."

## **ANALYSIS AND RECOMMENDATIONS**

Students are placed in internships and experiential situations with the primary goals

of learning and reinforcing academic experiences. Contributions from students are not always expected by the firm and costs to the firm are low. Costs of faculty internships, however, can be considerable, and the firm should expect benefits concomitant with its expenses. Care must therefore be taken in the planning and implementation of faculty internships.

### **Risk Management and Planning**

While faculty internships have risks, the risks are outweighed by the benefits to firms, faculty, and students. The risks can be reduced by planning and a clear understanding of expectations. Planning may involve consulting secondary sources about the firm as well as conducting preliminary interviews with executives of the firm. Interns must carefully assess their skills and fit these to the functions of the firm. Goals of the internship need to be clearly stated and the risks evaluated. Not all faculty will benefit from an internship; not all faculty should seek one.

### **Expectations**

First-time interns may feel awed by perceived expectations, and the firm may feel uncertain about how to use academic talent. Naturally, the intern must operate from present skills and talents, not expected ones. Later in the same internship or in a future internship, after the intern has gained experience, he or she can take on more diverse responsibilities.

What the intern delivers will be influenced by the organizational level and decision-making responsibility of the individual to whom the intern is assigned. If the intern reports to an individual with little or no independent decision-making authority, the

intern will most likely feel that his or her talents are wasted.

The person to whom the intern reports should understand the goals of the internship and appreciate the intern's skills. When there is ambiguity about goals, the intern risks being assigned tasks or to a mentor inappropriate to the intern's skills. For example, part of one internship involved training an employee in basic communication skills rather than in making contributions to corporate communication strategies. In this example, the corporate mentor was unclear about the capabilities of the intern and was not in a position to change the internship focus.

The intern must be prepared and realistic about his or her capabilities. This is not to say that the intern should not take risks, only that these risks should be manageable. Above all, the intern ought to be reasonably able to articulate his or her capabilities and to deliver what is promised.

## CONCLUSION

With proper planning and a realistic assessment of the prospective intern's talent and the sponsoring firm's needs, faculty internships can be immensely rewarding for all concerned. Contributions to the firm's well being can be made. Faculty gain experience and financial compensation, businesses gain new insights into their operations, and the academic and business communities gain a better understanding of each other. Finally, and perhaps most importantly, faculty develop a clearer sense of the "real world" in which some of their students already work and into which the rest will soon be entering. Class discussions are enlivened and made more cogent through faculty use of their own

experiences which exemplify course content. Also, experienced faculty are much better able to evaluate textbook information and their own cherished assumptions against what is happening in the practitioner's world.

## REFERENCES

- Cheatham, C. (1986). "Hire the management accounting professor," *Management Accounting*, p. 62.
- Krogstad, J. (1981). "The faculty residency: A concept worth considering," *Journal of Accountancy*, pp. 74-86.
- Lee P. (1988). "Faculty internship in public accounting: Planning is the most important step in implementation," *The CPA Journal*, pp. 84-86.
- Spinks, N. (1989). "Internships: Real world 101," *Journal of Education for Business*, pp. 15-17.

THE IMPACT OF CASE TOOLS ON ACHIEVING CRITICAL SUCCESS  
FACTORS IN SYSTEMS DEVELOPMENT

Mary Sumner  
Management Information Systems  
Southern Illinois University at Edwardsville  
Campus Box 1106, Building II  
Edwardsville, IL 62026-1106  
(618) 692-2504

ABSTRACT

The purpose of this study is to examine two questions (1) What are the "important activities" in systems development that deserve support? and (2) Do computer-assisted software engineering tools support these activities well? The findings illustrated a discontinuity between the use of CASE and the achievement of critical success factors in systems development. In the view of the CASE users, CASE tools did not enable them to achieve the "most important" critical success factors in systems development, such as the "ability to understand the client's business" and the "the ability to involve the client in the development process."

The introduction of computer-assisted software engineering (CASE) technology has caused many information systems professionals to assume that new tools will bring about significant increases in systems development productivity. CASE is viewed as a strategy to reduce development time, to cut maintenance costs, and to improve the discipline of information systems development.

Computer-assisted software engineering is often compared to the introduction of computer-assisted design and manufacturing (CAD/CAM) into the manufacturing process. CASE tools are used to facilitate greater standardization of work procedures and adherence with design discipline (Orlikowski, 1988).

Yet, even with its many benefits, most organizations have found it difficult to implement CASE. Some of the obstacles to introducing CASE are cost, resistance by systems developers, and unacceptable learning curve. Resistance by systems developers themselves remains an issue that is much more difficult to address than the quality of the tools themselves (Bouldin, 1987). Experienced designers and programmers feel that highly-structured CASE tools interfere with their job autonomy and creativity (Kull, 1987).

One of the issues relevant to the acceptance of computer-assisted software engineering tools is organizational fit. Success of an information system, Turner notes, is likely to be a function of the fit between critical organizational variables and the characteristics of the information system (Turner, 1982). Information systems implementation involves behavioral factors, and these factors are critical to the effective integration of technologies such as CASE.

One of the factors underlying resistance to CASE may be the mis-match between the work methods being

supported by CASE and the existing patterns that already exist within the setting of information systems development. As Markus points out, people resist information systems because of the interaction of specific technical system design features with the social context in which systems are used. Since new information systems may prescribe roles which are at variance with existing ones, they may create patterns of work and interaction which are at odds with the prevailing organizational culture (Markus, 1983).

Strategies such as the socio-technical systems design approach capture both the technical and social aspects of systems design. The socio-technical systems approach views organizations as made up of work systems that contain both technological and social aspects. The design of a work system must optimize both these technological and social aspects (Bostrom and Heinen, 1977). The technology of a work system includes the tools, the work methods, and the physical conditions for work. The social aspects consist of the roles fulfilled by people and the interactions among these roles. The technological and social aspects of a work system interact dynamically.

In designing a work system, one must optimize the technological and social aspects jointly. Two types of criteria apply in judging the quality of an effective work system. The first of these is task accomplishment, as measured by the quality of output, productivity, and reliability. The second criteria is quality of work life. The level of task identify, skill variety, job autonomy, and feedback are all factors which determine the quality of work life.

The obvious question about CASE tools is whether they effectively support the work system of the information system designer. In so doing, CASE tools should support the achievement of technical processes in



information systems design. For CASE to be effective, it must support both technological and social processes. Perhaps one of the reasons why CASE tools have not gained widespread acceptance is that they do not support these critical activities.

#### PURPOSE OF THE STUDY

The purpose of this study is to examine two questions: (1) What are the Critical Success Factors in information systems analysis and design; and (2) Do CASE tools support these Critical Success Factors? In John Rockart's definition, a Critical Success Factor is something "that must go right" (Rockart, 1979). If CASE tools support the most critical systems development tasks, then they should be used.

On the other hand, if CASE tools only support less-important activities in systems development, or if they make important ones more difficult to accomplish, they may be perceived by potential users as not valuable, which would lead to slow or no adoption by systems developers.

#### PROCEDURES FOR THE STUDY

This study required a series of steps. First, systems development experts were asked to identify a representative set of important activities in systems development, using the Critical Success Factors approach. Next, these systems development experts assessed the importance of each CSF and the degree of difficulty in achieving each CSF. After this, the researchers estimated the "need for support" for efforts to achieve each CSF, based upon its importance and difficulty.

Finally, a group of experienced CASE tool users judged the extent to which their CASE tools supported efforts to achieve each Critical Success Factor. This addressed the question of whether the CASE tools were supporting the critical activities in information systems development.

#### IDENTIFICATION OF CRITICAL SUCCESS FACTORS

The first step in the study was to identify Critical Success Factors in good requirements analysis, design, detailed design, and implementation. A Critical Success Factor is defined as "what must go right" to achieve successful results. Eighty-eight members of the CASE Users' Group in St. Louis were asked to respond to an open-ended questionnaire by listing what they believed to be Critical Success Factors in systems development. See Appendix A for the open-ended questionnaire.

Of the 88 members of the CASE users group, 28 responded to the questionnaire. These 28 respondents represented a cross-section of industries.

See Table 1

Each of these 28 respondents listed Critical Success Factors in requirements analysis, general design, detailed design, and implementation.

After these open-ended responses were analyzed, a list of most-frequently mentioned CSF's was derived. This list of CSF's was used to develop a questionnaire which was designed to assess the Importance and Degree of Difficulty of achieving each of these CSF's.

#### IMPORTANCE AND DIFFICULTY OF CSF'S

Eighty-eight members of the CASE Users' Group were contacted a second time to obtain responses to the Critical Success Factors in Systems Development Questionnaire (See Appendix B). Of this group, 26 members responded to the questionnaire. These respondents represented a cross-section of systems development organizations, including the consulting, manufacturing, and government environments.

See Table 2

Each respondent assessed the Importance of achieving each of these CSF's using a scale in which 5 meant "Must Achieve," 3 was "Very Helpful to Achieve," and 1 was "Not Critical." Each respondent also ranked the competencies in terms of Degree of Difficulty, using a scale from 6 (Most Difficult) to 1 (Easiest to Achieve).

As you can see from Table 3, two of the competencies in requirements analysis which were both "important" and "difficult to achieve" were the "Ability to involve the client in the development process" and the "Ability to set the boundary (scope) of a project." Although the "Ability to obtain support for the project" was considered very important, its degree of difficulty was less significant than the other two competencies.

See Table 3

In terms of the competencies in Systems Design, the respondents identified the "Ability to understand the client's business" as the most important and most-difficult-to-achieve CSF. See Table 4:

See Table 4

In Table 5, you will find the competencies associated with Detailed System Design. The most important of these competencies was the "Ability to establish effective communications between the designer and the user." The most-difficult-to-achieve competency was the "Ability to coordinate project activities so that tasks are completed within time and cost constraints."

See Table 5

Finally, the CSF's associated with implementation were evaluated in terms of their Importance and Degree of Difficulty. The "Ability to obtain customer acceptance of the final product" was considered to be the most important competency. The most-difficult-to-achieve CSF was the "Ability to manage the process of organizational change," as shown in Table 6:

See Table 6

#### VALIDATION OF THE SYSTEMS DEVELOPMENT COMPETENCIES

After the first questionnaire, each of the respondents received a follow-up questionnaire in which they were asked to evaluate whether the ranking of each of the Critical Success Factors was correct with respect to its Importance and its Difficulty. A five-point scale, ranging from "extremely well" to "extremely poorly" was used to assess the validity of the original rankings. See Appendix C for a copy of the questionnaire.

The results showed that the respondents agreed with these rankings. At this phase, the respondents were also asked to identify any "new" or "missing" factors which they considered to be essential for successful systems development and to create an Importance and Difficulty ranking for each of these new factors. The respondents were satisfied with the original listing of factors and did not introduce new ones into the existing ranking of these factors.

#### SUPPORT REQUIREMENTS FOR CRITICAL SUCCESS FACTORS

As the last step in assessing the critical success factors, the respondents were asked to determine the "support requirements" for each CSF. "Need for support" was seen as central to this study, combining CSF importance with difficulty. One CSF can need more support than another equally important CSF if it is more difficult to achieve. Conversely, one CSF can need more support than another equally difficult one if it is more important. What emerged from the results was a listing of the "most critical" and "least critical" success factors. Both the raw scores and normed scores are shown in Tables 7 and 8:

See Table 7

As you can see from these results, many of the "most critical" CSF's in systems development were related to effective communications with the user and to the ability to understand the user's requirements. This is consistent with prior research studies which also stress the importance of communications and interaction with the user.

The listing of "less critical" systems development factors (See Table 8) includes some of the technical

tasks in systems work, such as developing documentation, prototyping screens, and developing logical models:

See Table 8

The relatively low "need for support" of many of the technical aspects of systems design is consistent with the overall importance of analytical, problem-solving, and communications skills -- particularly during the requirements analysis and general design phases of a project. Experienced systems analysts recognized that without good user requirements, an effective technical design will not serve the overall business needs which a project is designed to address.

#### RATINGS OF CASE TOOL SUPPORT FOR CRITICAL SUCCESS FACTORS

The estimates obtained about support requirements for CSF's were used to derive a survey instrument to assess the extent to which CASE tools support the achievement of the CSF's. See Appendix D for the survey. The respondents to this final iteration of the study were all experienced CASE users. The members of the original survey group, the CASE Users' Group, volunteered the names of experienced CASE users within their respective organizations. In total, sixty-six CASE users were identified. Of these, twenty CASE users responded to the questionnaire. This response rate was considered quite good because many of the CASE users who were identified were "newcomers" to CASE and did not feel that they had enough experience to assess its impact with respect to the various systems development factors in the study.

The respondents represented a group of CASE users with experience using CASE tools in the context of a systems development project, ranging from requirements analysis to detailed design and implementation. This final group of respondents, the actual CASE users, represented a variety of organizations. See Table 9:

See Table 9

As you can see from this table, the majority of CASE users represented organizations in financial services, consulting, and government. These organizations are consistent with the types of organizations represented by the two previous samples of respondents who were responsible for identifying the Critical Success Factors in systems development.

The scale used to assess the extent to which CASE tools facilitated the achievement of Critical Success Factors ranged from strongly agree (+2), agree (+1), neither agree nor disagree (0), to disagree (-1) and strongly disagree (-2). As you can see from Table 10, the CASE users did not feel that CASE tools enabled them

to achieve the "most important" critical success factors in systems development:

See Table 10

CASE users did not view CASE tools as having a positive impact on the achievement of the most critical factors in successful systems development. They reported that CASE neither made it more difficult nor made it less difficult to achieve these factors. In addition, they agreed that CASE may make it more difficult to "involve the client in the systems development process" and to "establish effective communications between the user and designer."

Without additional information, it may be difficult to speculate on the reasons for these results. CASE was clearly not supporting the achievement of highly-ranked critical success factors in systems development. Perhaps the respondents were reacting to the highly-structured toolkits associated with certain CASE environments. By conforming to the tools and techniques prescribed by a CASE tool, they may feel that their creativity is limited. By investing time and effort in the use of CASE, they may feel that they have less time to interact with the user.

The respondents did not report that the CASE tools they were using made it easier for them to achieve some of the lower-ranked critical success factors in systems development. As you can see from Table 11, CASE had a neutral impact on the achievement of these factors:

See Table 11

The use of CASE did not make it easier to do such activities as generating design documentation, creating modular program design specifications, and designing training. CASE had a negative impact upon the ability to generate prototypes and the ability to develop a "good" logical model of a system. This is surprising because most CASE tools have data flow diagramming and entity-relationship diagramming tools which help the analyst derive process and data models for the existing system. Either these tools were not being used or else they were difficult to implement.

A similar comment can be made about prototyping. Whereas most CASE toolkits support prototyping of screens and reports, the respondents either found it difficult to use CASE to accomplish prototyping or else relied on tools (e.g. 4GL's, etc.) outside the CASE environment to do prototyping. They did not find that CASE made it easier to support the prototyping approach.

These findings illustrate a discontinuity between the use of CASE and the achievement of critical success factors in systems development. The results raise two important issues. The first is whether CASE is simply another technology which automates a series of isolated systems design tasks, without addressing the underlying need for "improving the systems development process."

CASE may be a technology solution which does not address the critical problem: how to design information systems which meet users' needs.

The second issue is: How can CASE be used to truly support the systems development process? Will existing CASE tools need to be modified to support the critical success factors in systems analysis and design more effectively? Or, will systems developers need to align their activities with the tools and techniques which CASE is able to support?

The answer to the seeming mis-match between CASE and systems development activities which experienced designers seek to accomplish may be made more clear through an examination of the socio-technical systems perspective. With an understanding of the STS perspective, it may be possible to integrate the use of CASE into the systems development process and to achieve the productivity gains which should result.

#### THE PROBLEM WITH CASE

The problem associated with introducing CASE may be partly explained by the socio-technical systems theory described earlier. Too often, CASE technology is superimposed upon an existing work system with no thought being given to current technical processes, work roles, and social aspects. In some situations, CASE is introduced into a systems development organization in which information systems design processes are not clearly understood or well-disciplined.

For CASE to be effective, an organization may need to view information systems development as a work system. That is, the critical processes of information systems development should be identified. The listing of critical success factors in this study might prove to be a starting-point for the tasks which must be achieved.

A social analysis should also be done. What are the roles of analysts and users in information systems design processes? What kind of communication is critical to accomplishing design tasks? How can the roles and responsibilities of analysts and users be clarified? What aspects of the work system are important for motivating the design professionals? What aspects are essential for gaining the involvement and acceptance of clients?

The alternatives to be evaluated should optimize the effectiveness of the technological and the social aspects of the proposed information systems development work system. At present, the use of CASE tools may facilitate the achievement of some of the technical aspects of systems design. However, for the information systems design work system to be effective, CASE tools must also facilitate social interaction and user involvement.

An ultimate design for an information systems development work system should be judged on the basis of the extent to which it achieves both task-accomplishment

and quality-of-work-life objectives. The ultimate design may incorporate the use of CASE tools along with the re-design of communication processes to facilitate client involvement. An over-emphasis on the technical aspects of systems design, without equal attention being paid to its social aspects, will not improve the effectiveness of the work system.

Perhaps one of the mistakes being made in the CASE environment today is the tendency to introduce CASE tools as a panacea. CASE, on its own, cannot improve the productivity of the systems development process. CASE can be used to support certain aspects of a new work system. The new work system for systems development must be designed with a thorough understanding of improvements in the technical process of design, and avoid the tendency to super-impose new tools and techniques upon antiquated work methods and procedures. The new work system must maximize the achievement of critical success factors.

#### IMPLICATIONS FOR PRACTICE

The findings of this study illustrate that CASE tools are not supporting the Critical Success Factors in systems development. One of the reasons for this is that CASE tools are being superimposed upon work systems in which antiquated methods are being used. Before CASE tools are introduced, we need to re-engineer the current work system so that critical technical and social processes are an integral part of systems development. In other words, we must re-engineer the current work system using the socio-technical systems approach.

Re-engineering the current work system will mean introducing structured methods using process and data modeling techniques in requirements analysis, design, and system implementation. Once systems designers become familiar with underlying structured systems design methodologies, it will be easier to introduce CASE.

The real impact of CASE is not limited to improving the productivity of the systems development work system. CASE must enable the systems designer to improve the work system of the user. CASE creates an opportunity to better understand the technical and social processes of the work system of the user and to re-engineer these processes.

As many companies move toward "total quality management" as an overall organizational strategy, CASE may facilitate the role of MIS professionals in business process re-engineering and process re-design. The use of process and data modeling methodologies in requirements analysis will enable designers to re-think existing work systems rather than to automate existing processes. CASE tools will also enable systems designers to create logical data models which specify requirements for shared databases. These shared data bases will support cross-functional work systems.

In conclusion, CASE will not be effective if it is super-imposed upon current work systems in systems development. These work systems must be re-designed to incorporate both technical and social analysis methodologies. Once these underlying processes are introduced, it will be easier for system designers to use CASE to make the work system of the user more effective.

#### REFERENCES

- Boland, R. J., "The Process and Product of System Design," Management Science, Vol. 24, No. 9, May, 1978, pp. 887-898.
- Bostrom, R. P. and Heinen, J. S., "MIS Problems and Failures: A Sociotechnical Systems Perspective." MIS Quarterly. Vol. 1, No. 3, September, 1977, pp. 17-32.
- Bouldin, Barbara. "Implementing CASE: From Strategy to Reality," Computerworld, Nov. 9, 1987, p. 518.
- Cheney, Paul H. and Lyons, Norman R., "Information Systems Skill Requirements: A Survey," MIS Quarterly, Vol. 4, No. 1, March 1980, pp.35-43.
- DeBrabander, B. and Thiers, G., "Successful Information Systems Development in Relation to Situational Factors Which Affect Effective Communication Between MIS-Users and EDP Specialists," Management Science, Vol. 30, No. 2, February, 1984, pp. 137-155.
- Green, Gary I., "Perceived Importance of Systems' Analysts' Job Skills, Roles, and Non-Salary Incentives," MIS Quarterly, June, 1989, pp. 115-133.
- Kaiser, K. and Srinivasan, A., "User-Analyst Differences: An Empirical Investigation of Attitudes Related to Systems Development," Academy of Management Journal, Vol. 25, No. 3, September, 1982, pp. 630-646.
- Keen, Peter, "Information Systems and Organizational Change," Communications of the ACM, V. 24, No. 1, January 1981, pp. 24-31.
- Kull, David. "The Rough Road to Productivity," Computer Decision, February 23, 1987, pp. 30-41.
- Markus, M. Lynne, "Power, Politics and MIS Implementation," Communications of the ACM, V. 26, No. 6, June 1983, pp. 430-444.
- Orlikowski, Wanda J., "CASE Tools and the IS Workplace," Proceedings of the 1988 ACM SIGCPR Conference on the Management of Information Systems Personnel, College Park, Md., April 7-8, pp. 88-97.

Rockart, John, "Chief Executives Define their Data Needs," Harvard Business Review, March-April 1979.

Turner, Jon, "Observations on the Use of Behavioral Models in Information Systems Research and Practice," Information and Management, V. 5, 1982, pp. 207-213.

Vitalari, Nicholas. "Knowledge as a Basis for Expertise in Systems Analysis: An Empirical Study," MIS Quarterly, Vol. 9, No. 3, September 1, 1985, pp. 221-241.

Table 1: CASE Users' Group Respondents

<u>Type of Organization</u>	<u>No. of Respondents</u>
Government agencies	10
Insurance and financial	8
Consulting and contract development	5
Utilities	3
Manufacturing	2
Computer vendor	1
Total	28

Table 2: Respondents to the CSF Questionnaire

<u>Type of Industry</u>	<u>No. of Respondents</u>
Insurance and financial	8
Government	7
Manufacturing	5
Consulting, contract development	3
Utilities	2
Computer vendor	2
Retail	1
Total	26

Table 3: Competencies in Requirements Analysis

	Degree of Importance	Difficulty
Ability to involve the client in the development process	4.77	4.15
Ability to obtain support for the project (time/resources)	4.85	3.54
Ability to set the boundary (scope) of a project	4.38	4.42
Ability to identify the problem/opportunity within the boundary of a project	4.23	3.54
Ability to decide whether it will be worthwhile to pursue solution of the problem/opportunity	3.92	3.12
Ability to choose the team who will do investigation and modeling	3.23	3.00

Table 4: Competencies in Systems Design

	Importance	Degree of Difficulty
Ability to understand the client's business	4.54	4.35
Ability to communicate the results of investigation and modeling activities to those who must approve them	4.46	3.96
Ability to investigate the existing system, its environment, and its functions	3.46	3.23
Ability to create alternate "good" logical models to represent possible solutions to problem/opportunity	3.77	3.73
Ability to produce a "good" logical model (i.e. consistent, complete, valid, flexible) of the existing system	3.23	3.62

Table 5: Competencies in Detailed System Design

	Importance	Degree of Difficulty
Ability to establish effective communications between the designer and user	4.61	3.92
Ability to coordinate project activities so that tasks are completed within time and cost constraints	3.92	5.04
Ability to document systems design specifications accurately and completely	3.92	3.42
Ability to create modular, flexible program design specifications	3.77	3.65
Ability to construct a simple, effective user interface in the design of reports and screens	3.69	3.19
Ability to prototype the design of reports and screens so that user requirements are defined	3.38	3.00

Table 6: Competencies in Systems Implementation

	Importance	Degree of Difficulty
Ability to obtain customer acceptance of the final product	4.85	3.38
Ability to maintain effective communications between the analyst and user	4.46	3.69
Ability to develop and implement an effective training program	4.31	3.27
Ability to design and implement effective testing strategies	4.15	4.04
Ability to manage the process of organizational change	3.92	4.65
Ability to develop thorough systems design documentation	3.04	2.73

Table 7: Most Critical Systems Development Factors

	Raw Score	Normed Score
Ability to obtain support for the project	4.76	4.99
Ability to understand the client's business	4.53	4.53
Ability to obtain customer acceptance of the final product	4.47	4.48
Ability to involve the client in the development process	4.47	4.34
Ability to maintain effective communications between the analyst and user	4.41	4.34
Ability to set the boundary (scope) of a project	4.35	4.32
Ability to establish effective communications between the designer and the user	4.29	4.14
Ability to coordinate project activities so that tasks are completed within time and cost constraints	4.18	3.93

Table 8: Less Critical Factors in Systems Development

	Raw Score	Normed Score
Ability to document systems design specifications accurately and completely	3.18	2.14
Ability to prototype the design of reports and screens so that user requirements are defined	3.06	2.10
Ability to develop and implement an effective training program	3.17	2.08
Ability to create modular, flexible program design specifications	3.06	2.03
Ability to choose the team who will do investigation and modeling	3.00	1.86
Ability to produce a "good" logical model of the existing system	2.71	1.59
Ability to develop thorough systems design documentation	2.65	1.40
Ability to create alternative "good" logical models to represent possible solutions to problem/opportunity	2.47	1.01

Table 9: Characteristics of CASE Users

<u>Type of Organization</u>	<u>No. of Respondents</u>
Insurance and financial	8
Consulting, contract software	5
Government	4
Manufacturing	2
Defense	1
Total	20

Table 10: Impact of CASE on "Most Critical" Factors

	Raw Score	Mean Impact Score
Ability to obtain support for the project	4.76	0.0
Ability to understand the client's business	4.53	0.0
Ability to obtain customer acceptance of the final product	4.47	0.0
Ability to involve the client in the development process	4.47	-1.0
Ability to maintain effective communications between the analyst and user	4.41	0.0
Ability to set the boundary (scope) of a project	4.35	0.0
Ability to establish effective communications between the designer and the user	4.29	-1.0
Ability to coordinate project activities so that tasks are completed within time and cost constraints	4.18	0.0

Table 11: Impact of CASE on Less Critical Factors

	Raw Score	Mean Impact Score
Ability to document systems design specifications accurately and completely	3.18	0.0
Ability to prototype the design of reports and screens so that user requirements are defined	3.06	-1.0
Ability to develop and implement an effective training program	3.17	0.0
Ability to create modular, flexible program design specifications	3.06	0.0
Ability to choose the team who will do investigation and modeling	3.00	0.0
Ability to produce a "good" logical model of the existing system	2.71	-1.0
Ability to develop thorough systems design documentation	2.65	0.0
Ability to create alternative "good" logical models to represent possible solutions to problem/opportunity	2.47	0.0



## DATA COMMUNICATIONS: A LOOK AT CONTENT REVISION

Gerald J. Tatar, A. J. Palumbo School of Business, Duquesne University, Pittsburgh, PA 15282 (412) 396-6266

### Abstract

This research paper presents the findings of the writer in addressing content revision in data communications. With the current emphasis on networking in both the business sector and academic institutions it is necessary to address this concern in presenting the content in a data communications course. Views and actual findings are presented in regard to Novell network installations.

### Introduction

In the field of MIS the only constant seems to be change. New hardware and software along with new end user applications drives the field to continually change with the technology. Many universities and businesses have installed networks to permit users to share resources and communicate. Along with these newly added capabilities comes added responsibilities to users and instructors of the technology. Traditionally the Data Communications course is an area in the MIS curriculum that addresses networks. Most recent college textbooks published in this area address networks in a chapter or less of reading material. It is time to enhance the students exposure to this growing field.

### Background

In a number of data communications books there are a range of topics covered. The ISO-OSI reference model is introduced. The telephone system is discussed from a national and international perspective. Other topics include telecommunication applications, voice communications, data terminals, coding and digitizing, circuits, data transmission, protocols, architectures and standards, telecommunications management, and many other topics. In addition to wide area networks the writer believes that local area networks need to be taught and discussed in much more detail to better prepare students for the challenges that await them in the work world. Carlton Amdahl, chairman and chief technology officer of NetFrame Systems, Inc., says "I personally believe that server technology is the ultimate replacement for the traditional centralized mainframe" [5]. In particular, more Novell LANs are installed at business sites and university sites across the country than other types of LANs. Novell netware has nearly become the defacto standard for LAN installations.

The writer installed a faculty Novell LAN over the last several years. This ArcNet LAN has grown to seventy nodes. The LAN has also been bridged to the university mainframe. Recently a new public personal computer lab has been installed and Novell networked giving students access to LAN tools. Similarly, a new multimedia lab has

been installed and the microcomputers will be Novell networked. Interconnected LANs are now replacing WANs [3].

The point here is that students at the writer's school have access to Novell netware and have a unique opportunity to learn new concepts that are not traditionally covered in an MIS course on data communications. When instruction is given on the network chapter, some content that could be added includes installation of network cards, resolution of memory contention problems, setting network card addresses, installation of cable connectors, and installation and setup of Novell netware on a file server. This could be presented to the student using a "practice LAN" providing the equipment was available.

### Software Utilities

Some of the Novell netware software utilities that could be covered include Syscon, Pconsole, Fconsole, Filer, Session, and others. A discussion of system accounts could uncover the thinking and planning that goes into account creation. This would include considerations such as password length, unique versus nonunique passwords, required password changes at periodic intervals, time and date limitations to access the system, disk space limitations, groups belonged to, rights to directories and files in those directories, virus considerations, and security considerations. A discussion of Login scripts with associated environment variables for particular software, trustee assignments, file rights such as read, write, open, create, delete, parental, modify, and search rights could be discussed. Menu design and menu programming should be presented. Special user needs could be addressed in the instructor's presentation of the material.

### Printing Concerns

Print queue administration should be designed and developed. Who will administer the print queue? Who has print job delete privileges? How will different printer types be dealt with such as a dot matrix printer and a laser

printer? How will printing on special forms such as letterhead and mail labels be done? These and many other printing considerations could be raised so that the student could be well trained in these areas.

### Other Content Matter

Supervisor status is another issue to be addressed. There should always be a trap door (i.e. an alternate supervisor account) so that the system administrator can access needed utilities in the event that the main supervisor password is lost or forgotten. Who will be the supervisor(s)? Can they be trusted not to abuse their privileges? These and other questions can be raised and resolved to get the students thinking.

In the event that the LAN has multiple file servers a number of other issues could be raised. What users need access to two or more servers? How are files moved between servers? How will data redundancy be dealt with? Who has physical access to the servers? Will uninterruptible power supplies need to be installed on the servers? What software will be installed on each server? How will the LAN be balanced to deal with special issues such as demanding resource contention? As the reader will note, the concepts learned from such an exercise could be transferred to setting up LANs of other types also.

### A Windows Environment

Working with Microsoft windows on a LAN can also present an interesting challenge for the students. What new services can be provided in the windows environment? The student could be taught how to multitask a mainframe session, a LAN session, and a local session. New printing problems will result and will have to be dealt with. In the writer's situation, a Novell product called LAN Workplace for DOS (LWP) was used to bridge the Novell LAN to the academic DEC VAX-8550 computer system using TCP/IP. "TCP/IP permits communication between diverse systems over a wide area without using a vendor-specific architecture" [1]. With a mainframe session running in a window, the student can access internet with all of its capabilities. University wide electronic mail is also feasible. The File Transfer Program (FTP) can be used to move files between the personal computer and the remote node. All of these new capabilities and others could be discussed in the data communications course when the LAN section is taught. With all of the corporate downsizing currently occurring, this knowledge will give the student a fighting chance to survive in today's world of MIS. A significant task exists for the MIS professional charged with moving mainframe applications to local area networks.

### Other Considerations

Cabling, active hubs, passive hubs, repeaters, routers, protocols (ArcNet, Ethernet, Token Ring, Proprietary) can all be introduced to the student. LAN topology should also be addressed in the LAN section of the data communications course. "The three most commonly used topologies are the ring, star, and bus" [2]. LAN maintenance should also be addressed.

### Summary

As the reader can see there is a lot of material that can be covered in the LAN section of a data communications course. "Unlike railroads, successful standardization of transportation lanes for the information age - networks - has not been realized" [4]. Future textbooks should take a proactive approach in communicating the above mentioned concepts in the LAN section of the course. It is time that the data communications course content be revised in the area of local area networks to better prepare today's student for working on local area networks.

### References

- [1] Haverty, Jack. "Using TCP/IP as an Interim Multivendor Solution", Networking Management, August, 1989, 42-44.
- [2] Kee, Robert and Leitch, AL. "Local Area Networks - Enhancing Microcomputer Productivity", CPA Journal, August, 1989, 16-23.
- [3] McQuillan, John M. "Routers as Building Blocks for Robust Internetworks", Data Communications, September, 1989, 28-33.
- [4] Mier, Edwin E. "LAN Gateways: Paths to Corporate Connectivity", Data Communications, August, 1989, 72-84.
- [5] Till, Johna. "Multilingual Server Dishes Up Data and Applications", Data Communications, October, 1989, 167.

# INTEGRATING INFORMATION SYSTEMS TECHNOLOGY IN THE ORGANIZATION OF THE FUTURE

Herman P. Hoplin  
School of Management  
Syracuse University

## Abstract

In projecting the ISECON '93 theme, "IS Education--The Future of Information Systems," leadership in managing the use of information technology comes to the forefront. The goal of this paper is to expose and present those issues which can be drawn together to integrate IS technology in organizations of the future.

## INTRODUCTORY BACKGROUND

Over the short history of computer-based information systems, professionals have gone through a series of generations which were initially based on heavy technical development of hardware, soon followed by software, and later by peopleware. Early developments were focused on programming and data processing. Later, preoccupation with getting the computer to work gave way to concerns about what functions computer systems should do and numerous applications in which high technology would serve business as well as research and development operations.

Information systems soon became a feature of automated systems where word processing, decision-support and decentralized processing developed into major courses of action in carrying out business functions. The digital computer opened the door to replacing electrical accounting machines (EAM) and in a generation or two this led to office automation.

Greater demand for applications led to the need for integrated systems such as computer-based management information systems (MIS). This era began in the 1970's. The system (MIS) was based on the idea of serving management at all levels and demanded an integrated communications (reporting) component. It soon became apparent that the technology available at that time would not support the MIS vision of a single computer-based information system to meet the needs of an organization for transaction processing, financial and control reporting, manufacturing planning and control, etc. (Hoffman, AMA FORUM, September 1991)

This led to the next component of the MIS or cycle of the MIS process -- Data Base Management

System (DBMS). This DBMS concept vastly improved the prospects of having a single, comprehensive database where all data could be deposited to meet the needs for application processing and answering ad hoc questions for management. Much as this improved the concept of an MIS, it fell short due to the resource requirements, cost problems, and it barely scratched the surface on meeting decision-support requirements of the organization.

About the time that DBMS technology was ready to stretch out into the area of distributed DBMS, PCs began arriving on the scene in droves. The tight organization that linked all programs to the mainframe became threatened. The established systems based on the mainframe were challenged and soon relational-type DBMS were on the market for use with PCs.

As some authors have found, it is useful to consider IS as a business within a business even though integrating IS into the other functional areas of the firm may cause special organizational and strategy-formulating challenges. (McFarlan and McKenney, 1983) However, this integrated approach must be taken in order to address the organizational issues surrounding IS. This paper focuses on this approach which is drastically changing the nature of the IS field. Notable among these trends are the use of IS technology as a part of corporate strategy, end user computing, and the use of PC's as managerial workstations. (Lucas, 1990)

## METHODOLOGIES AND ORGANIZATIONAL ISSUES

The impact of computer supported technology on information systems development may be much greater than either the computer professionals or top and functional managers perceive it to be. As several co-

authors state, professionals find themselves in transition from technicians to managers. (Amoroso, et al, 1989) Unfortunately, many professionals have not learned how to manage IS.

In this environment of rapid change, new technologies need to be grasped and introduced into IS development or use as management tools in all the management processes; however, the enormity of this task is such that the success of technological change is not foreordained. Here is where professionals, users, and functional managers must work as a team to bring the desired results. This type of planned change now requires the use of both managerial and technical skills and enlists the use of Computer-aided Software Engineering (CASE) tools in order to accomplish desired and lasting results.

This paper reviews a number of technical methodologies and innovations that can be harnessed to enable advances in IS development including the strategies and interfaces needed to effect organizational change. This is a multi-disciplinary, and in some instances multinational, effort requiring an open systems approach in a hitherto technical domain which could not, by itself, successfully contain the rapidly spreading information technology. This calls for some new directions in IS structure such as using IS technology in corporate strategy formulation to gain a competitive advantage.

**Methodologies** Methodologies worthy of note for integrating IS technology are discussed briefly in this section.

Jones and Kydd discuss the use of A Systems Development Methodology across organizations. Their basis for the use of ASDM is "that organizations process information to reduce uncertainty and to resolve equivocality." (Jones and Kydd, 1988) In this study, they evaluated the use of ASDM in two different companies. Organization A did not receive it well due to mixed messages from management. The perceptions of the employees were that management was not totally committed to the ASDM. Organization B, on the other hand, was able to implement ASDM and use it to their advantage.

The implications of this study are that systems development methodologies do not transfer easily across organizational boundaries. The same methodology that works for one company will not necessarily work for another. This could be due to

organizational intricacies such as structure or internal power. A second implication is that the involvement of senior management must be perceived as genuine. When there are mixed signals, the project meets with resistance.

Another methodology more frequently used in systems development is the CSF method, or critical success factors. "The CSF methodology is a procedure that attempts to make explicit those few key areas that dictate managerial or organizational success." (Rockart, 1979) It requires that the analyst or consultant have a strong understanding of the business. That understanding is then used to elicit information from executives regarding critical success factors for the organization. These CSF's can be used to promote organizational redesign issues and initiate strategic planning. Simultaneously, CSF's can be used to define information technology capabilities for the organization and eventually link the organizational and information needs.

This methodology encourages communication throughout the organization. Rockart cautions that this process will not work at all times in all companies. Timing is key. Management must be ready to be involved. Competitive pressures or sheer awareness of the increasing strategic importance of information systems are among a long list of factors on which the outcome of the process is dependent. (Rockart and Crescenzi, 1984)

CSF methodology has a place in information systems development. Depending on the organization, it can have great strategic implications. The eliciting of critical success factors serves two purposes. The first to help management identify those issues that are increasingly important to the organization. Second, is that it gets management actively involved. It is not only a systems development effort; rather, it is an organizational development technique. It has far reaching consequences for integrating organizational and IS goals.

Other potentially valuable methodologies for integration are:

- (1) Ends/Means Analysis (MIS Research Center, Univ. of Minnesota)
- (2) Business Systems Planning (BSP--IBM)
- (3) Nolan's Stage Analysis
- (4) Andersen Consulting--M-1

## Organizational Issues

Methodologies operate within the context of an organization. In order to determine the appropriate methodology, we must consider certain organizational issues. These consist of the organization structure, culture (power and politics in the organization), and the organizational position of Information Systems (IS).

Organizations can be categorized by their structure: functional, divisional, matrix, centralized and decentralized. (Markus, 1983) This structure will determine the official lines of control in the organization. This will, in turn, determine the type of technology to use and its appropriate implementation strategy. IS systems professionals need a full understanding the structure of the firm in which they operate. They must also have an understanding of how certain technologies and implementation strategies can best fit that organizational structure.

Communications play a similar role in organizational IS. A model for evaluating communication patterns in an organization as a means of determining the current organizational structure was developed. (Crowston, et al, 1987-88) In this model, they examine the link between the structure of the organization and the information technology being used. The outcome identifies the implicit structure, or internal power in the organization. Provan discusses the impact of internal organizational power on strategy decisions. Due to the far reaching consequences of technological change, it is reasonable to consider information systems development as a strategy decision. He defines power as the ability of a department to influence the formulation and implementation of strategic-level decisions in a way that is supportive of its needs and perspective. (Provan, 1989) By adopting this definition, it becomes obvious that a department in good standing within the organization could have ultimate control over IS development decisions.

The strategic decision process is influenced by major groups in the organization with the most power. This suggests that the impact of external factors may be diminished, and that strategy decisions are based on the internal factors at work in an organization. However, power and tradition may play a more direct role in the allocation of resources and in the formulation of strategic decisions. The IS manager must be aware of both the internal and external issues and attempt to reduce the impact on the IS organization. Consideration must be based on the organization as a whole, not just suboptimization of the best department.

Many problems arise due to the fact that IS has not been in the organization as long as other departments. It does not have the tradition behind it. This internal focus could cause an organization to overlook certain technological advances that might enhance their competitive position.

Another factor to consider is the organizational position of IS. This position can determine the extent to which information resources are utilized. If IS is considered a support department, the typical areas will be automated: accounting, payroll, purchasing, etc. These areas require simple data manipulation which requires primarily technical skills in order to convert existing practices into automated ones. If, however, IS is in a strategic position in the organization, automated decision making, or Decision Support Systems, usually results. This requires systems professionals to be versed in business processes and have good communication skills in order to automate these major management cognitive processes.

The position of Information Systems changes in an organization as time passes. As technology is integrated into the organizational framework, the emergence of IS assumes a more strategic role. As the role of IS changes, so will the skills required by the IS professionals involved in the operation. This will not happen, however, unless organizations change internally and accept IS as an important link in strategy and decision making.

## **PROLIFERATION OF INFORMATION MANAGEMENT**

Without fully realizing the rapid pace of change, professionals are faced not only with developing the connectivity between mainframe and PC systems but are now having to deal with another area of data communications sometimes referred to as Networking. In an article in the Harvard Business Review on "How Networks Reshape Organizations--For Results," the point is made that "a network identifies the 'small company inside the large company' and empowers it to lead." (Charan, HBR, Sep/Oct 1991) No longer is communications a separate field from data processing; it now is a part of distributed processing. Networks began to matter when they changed the behavior of the attitudes and dialogues among managers. Data processing personnel and MIS managers found that Local Area Networks (LANS), Wide Area Networks (WANS), satellites, special hardware and software technology, and protocols are

now needed for integration with the MIS. This calls for close coordination and a requirement for new architectures. This is basically where we are today. More on architecture will be covered later in this paper.

The introduction of communications into information systems did not take place in an environment of concentration in that area. To the contrary, other developments were transpiring. Two major ones are Object-Oriented Programming (OOP) and Expert Systems Technology (ES).

Both OOP and ES are major breakthroughs in high technology. The OOP concept is that objects containing both data and information to be processed can be moved about, used again, etc. (Hoffman, 1991) This looks at data from the user point of view and requires a new approach to the information systems approach that supports them. ES are essentially programs designed to emulate a human expert to improve the decision-making process. These systems change the thinking process and provide a window to fill voids in the conventional management process. ES when applied in the context with DBMS technology adds the missing element of logic and also requires a broadening of traditional architecture--resulting in the blending of ES and DBMS for management use.

### ENSUING PROBLEMS

The foregoing events have surfaced many significant problem areas which need to be solved and managed. To bring this in context, the observation begins with Nolan's stage analysis. In the author's view, Nolan accurately projects the development from the EDP era to the IS era (Figure 1 at end of paper).

As he did in his model, we should now be concentrating on managing the advanced stages of computer technology which are in the IS era. He points out that technological discontinuity occurs near the end of Stage III Control which is the critical point where preoccupation with the computer must give way to what the system can provide to the enterprise through integration, architecture, and distribution of the output. In this case this means choosing enterprise over entity, or the "big picture" over the technical details, in the focus of management's requirement for information for the organization.

#### Issues of Change and Conflict

There are human elements inherent in an

organization. Their impact on IS development can be categorized in many ways. The fast changing nature of technology requires looking into human and organizational reaction to change and conflict.

Markus points out that resistance is easiest to identify when a person behaves in a manner that may result in the disruption of a system that is used by others. (Markus, 1983) An example of this is an employee who continues to manually post accounts after an automated system has been introduced for the organization.

Resistance can be minimized if an organization preparing to undergo change keeps employees informed along the way. This has major implications for technological change. First, it follows a user-involvement approach. If the users are involved all the way, uncertainty will be decreased. They can also give input as to the least disruptive way to institute the change. Second, it emphasizes the need for the technological manager to be aware of personal issues involved in planning and changing information systems. If we combine this with what Markus says, technological change must be carefully planned to coincide with the direction of the organization and systems professionals must be prepared to deal with some amount of resistance and stress on the part of those affected.

As change must occur for an organization to survive, conflict will follow. Differing views, value systems, and perceptions can generate conflict. In viewing conflict in organizations, a distinction can be made between cooperative conflict and competitive conflict.

Equating conflict with opposing interests and competitive goals may impede the understanding and managing of conflict. Therefore, those who conclude that their goals are competitive have more difficulty in managing conflict. By believing that goals are cooperative, more significance can be placed on productive conflict management in organizations. (Tjosvald and Chia). This emphasizes the need for a common understanding of organizational goals.

What does this mean for IS? Essentially, it implies that information systems departments need to establish themselves in the organization in such a way that they support the common goals of the organization (consulting experience indicates that organizational goal orientation is often lacking). Occasionally the perception is that IS is in competition with the employees. The fear that technological change will

eradicate jobs, generates a great deal of conflict in organizations. As IS changes occur, they may naturally be met with conflict. However, if all the people involved are moving toward a common goal, the conflict is resolvable.

Both issues, change and conflict, could be supported by the CSF method mentioned earlier provided that the appropriate organizational structure exists. Not only does CSF identify common goals, it employs the entire organization in order to reach them. Through identifying the critical success factors, organizations can move to the future.

### Problems in Assimilating Technology

Technological assimilation into the organization is difficult. One reason is that IS technology has had a short life. It cannot be as well established in an organization as the financial functions are because it has not been around as long. Another reason may be that the field has undergone dramatic evolution in its technologies. What used to take years to go from research to use now can take minutes. With every new system installation, there are newer and better systems being designed. With the current exploration of expert systems, the potential becomes almost unlimited.

Another reason may be the complexities in IS development which force creation of specialized departments and strain relationships. These specialized departments require increased specialization and generate a greater need for skills from IS professionals. (McFarlan and McKenney, 1983)

Organizations are not structured to change as quickly as technology does. This could have strategic implications for the organization of the future as it attempts to assimilate technology into its core. Conversely, organizations that do not change structurally may not be able to take full advantage of technology.

### Executive Information Systems

There is a solution to many of the problems addressed here. This is to take a top-down approach and have an Executive Information System (EIS). EIS and MIS, when used together, can be powerful tools. MIS have integrated the functions of data reporting, data storage and retrieval, and decision support. EIS have the potential of supporting executive managers by pointing the way to problems in need of solutions.

There are many approaches leading to an EIS. One of those that many researchers favor and makes a lot of sense for the future is using the strategic MIS theory. Its goal is to strive for the strategic alignment of the firm and the establishment of the desired future computing environment, not necessarily in that order, in a long-range MIS plan. The three major steps involved in this approach are: 1) To tie the long-range MIS plan to the organization's long-range plan by examining the firm's mission statement, objectives and goals as well as technological developments that relate to these areas; 2) Include the technologies the firm wishes to develop in the future into the MIS long-range plan; 3) Once the above steps have been taken, the technological forecasts and organizational goals are combined based on interviews with users. The long-range goals of the MIS department are chosen based on the results of the interviews.

### **SYSTEM INTEGRATION THROUGH ARCHITECTURE**

If anything, data processing and the Computer Industry have been plagued with too many sudden changes and resultant abandoned and obsolete systems and architectures. Not only did the advent of the PCs result in near anarchy (Hoffman 1991) but the other changes going on have brought on requirements hitherto not planned for. This forced professionals into a cycle of architectural changes which left many carcasses scattered around the environment. More than one professional and functional manager has asked the question, "Where do we go from here."

Rapid changes in technology and in user requirements have challenged designers who are now beginning to recognize that design must be flexible and it cannot be frozen in-place without the risk of obsolescence. Designers are now recognizing the need to create an overall design in pieces, as it is needed, with each piece fitting into the overall design (Zachman, *IBM Systems Journal*, Vol. 26, No. 3, 1987, Hoffman 1991). To do otherwise results in waste and actually retards progress. The challenge now is to envision an architecture built on ideas to meet evolving needs and utilize the best of all worlds--the IS expanded system together with new ideas evolving in Expert Systems and the communications field such as protocols, standards, and cooperation in both vendor and International areas. This broader view also must be expanded to include artificial intelligence (ES) and Computer-Aided Software Engineering (CASE tools). Without the latter, development time lags far behind the realities of our current scene.

## THE FUTURE OF ORGANIZATIONS AND THE ROLE OF IS

As many in attendance at this conference will attest, managing organizational change appears to be the key to success in both large and small organizations. Researchers have found that even with an organization's survival at stake, getting an organization to change is a difficult task. Change in organizations primarily occurs because of external (environmental) pressure rather than an internal need to change. (Goodstein and Burke, 1991) Unless organizational change can be done successfully, the future of the organization will be short-lived.

In addition to the "change" challenge, management authorities see our organizations heading in different directions. Peter Drucker sees the organizations of tomorrow flattening out. There will be little if any hierarchy and the employees will become specialists, much like we see in a hospital. He sees information becoming more important and even calls it the information-based organization. There will be multiple factors causing this change, but Drucker believes that information technology will drive the shift. He discusses the shift from data processing to information processing and the resulting organizational structure changes. (Drucker, 1988) This viewpoint seems to highlight the importance of identifying critical success factors. Ralph Carlyle, on the other hand, believes that centralization will return and that, while there will be specialists, the "organization will be eager to spawn multidisciplinary generalists. (Carlyle, 1990) Carlyle recognizes the impact that Information Technology can have on the organization in that he believes that Information Technology will be an enabler of change and that the change will be a result of what business professionals want.

Although these views differ, they both may be correct. The implications of these differing views are multi-faceted. First, IS managers of the future will be faced with newer and more difficult challenges. These will include selecting the appropriate technology for an organization and being managers rather than technicians. Second, the future role of IS is ambiguous, at best. It may enable change or cause it. Whatever the case, as organizations evolve, so will the role of technology in the organization.

### The Role of IS

There are many theories of the role that

Information Systems should play in an organization. Gorry and Scott-Morton stated that totally integrated MIS are not necessarily cost efficient or supportive of organizational goals and the IS should exist essentially to support decision making for the type of decisions involved. (Gorry and Scott-Morton, 1989) This raises the issue of IS being a support department and requires IS professionals to be cognizant of the decisions to be made in an organization as well as be aware of the thought processes of the people who will be making the decisions.

Donovan suggests that the role of CIO (as the top information officer of the organization) will lead to network management and cause the need for setting technological and organizational ground rules. (Donovan, 1988) This can lead to user issues over resources and internal power. Under these conditions it is essential that IS professionals insure that all portions of the organization have access to the appropriate technology which may require a high degree of control.

IS managers are increasingly viewing their own roles and that of their departments as supportive in nature. More and more IS professionals are accepting and adapting to their changing roles in the organization. No matter where IS is placed or the role of its managers, systems professionals will continually be faced with organizational and human issues in the organization environment.

### Where Do We Go From Here?

In view of the impact on management, user functions, and the high technology function, a broad, encompassing, and flexible architecture is needed for information systems success. We are no longer living in the "good old days" of data processing. We now need a new architecture to bring together and manage as a coherent whole the integration of:

- (1) Data Processing
- (2) Word Processing
- (3) Business and Organization Systems
- (4) Distributed Processing including End-User Computing
- (5) Electronic Data Interchange (EDI)
- (6) Use of Automated Tools (CASE)
- (7) Expert Systems
- (8) DBMS to include OOP-type systems
- (9) Data Communications and Networking
- (10) Neural Systems
- (11) Virtual Reality



- (12) Windows
- (13) Other types of connectivity yet to be developed

It is now a mammoth task for managers and professionals of information systems to have an overall design goal where the pieces (elements or modules) of the puzzle can be inserted rapidly to meet shifting demands. Some progress has been made to develop the open systems approach, international standards, and more versatile operating systems. The challenge, however, is an imposing one. The architecture must not only be based upon the sensible application of the standards and protocols of the industry but respond to the current and future needs of the environment. This is an enormous and humbling task. It requires designing systems for people--not computers--and requires understanding the behavior of these systems and networks for performance evaluation and sustained success in the future. This requires our best shot and is a team effort.

#### IMPLICATIONS AND CONCLUSIONS

So, what does all this mean for the IS professional and the IS department?

(1) The IS manager is in a multi role, a professional transition moving from technician to manager with considerable exposure as a trainer and educator. There is a responsibility to educate the organization about the technology available and how it can support the organization's goals. IS professionals must educate management and users to increase their awareness of the strategic importance of IS.

(2) The fundamental problem is one of designing integration to influence management in the use of information technology to best accomplish and expedite the organization's goals and objectives by the use of automation (CASE tools), the use of multiuser design teams, decision support systems, end-user computing, expert systems, and, where applicable, the development of international and global systems.

(3) Technology provides new options since no two organizations are alike; this precludes the use of the same technology for design. Multiple tasks are needed to accommodate the complexities of various goals, different organizations' cultures, and myriad personal styles. Each organization must identify its own goals and the process through which they can be achieved. Systems professionals must be able to do this. The CSF methodology discussed in this paper is

an example of one strategy to accomplish this.

(4) Recognizing and accommodating human responses to change and conflict is also paramount. If common organizational goals are established, an environment of cooperation is more likely to exist. No matter what approach is taken, there will be resistance to change and some conflict. The role of the IS professional then becomes one of facilitator or catalyst as the organization evolves through the process.

(5) The organization's goals (business needs) must be considered as the basis for change and use technology options to accomplish this.

(6) As technology becomes more widespread in the organization, IS must involve users more and more. Not just end users, rather the entire organization.

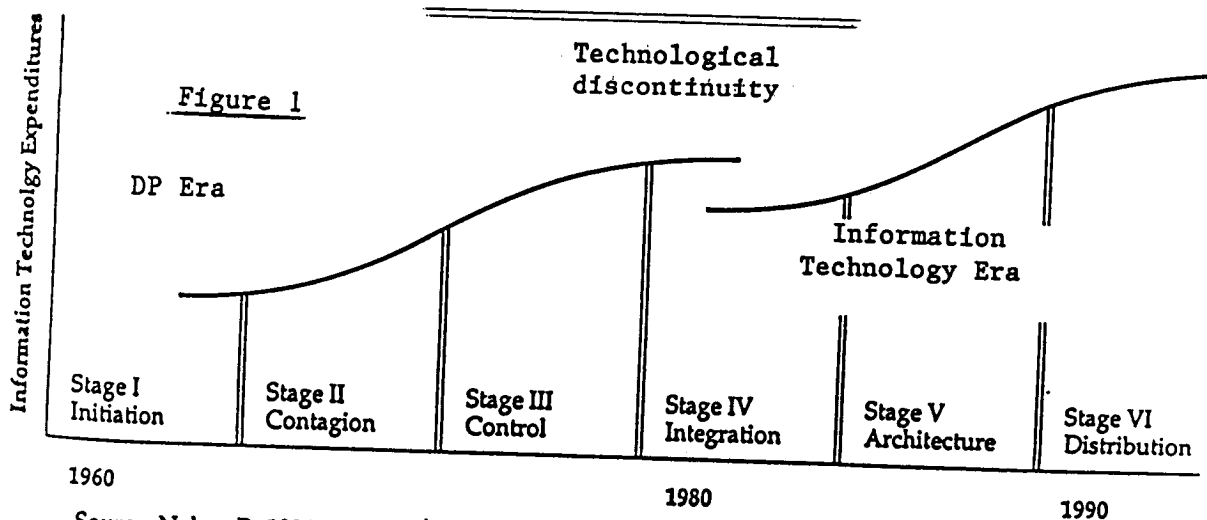
(7) Because IS is in a rapidly changing state (perhaps only the tip of the iceberg is now showing), much additional research is needed in this decade if we are to bring about success from introducing innovative computer-supported technologies.

Pessimists may feel on the surface that, "Alas, there is little progress in meeting the integration problem." However, now that we have received the "wake-up call," great strides are being taken in integrating IS driven by the forces discussed in this paper. We have no choice but to do better now and not extend the IS Odyssey until 2001.

#### References

- Amoroso, Donald L., Thompson, Ron L. and Cheney, Paul H. "Examining the Quality Role of IS Executives: A Study of IS Issues," Information and Management, 17 (1989) pp. 1-12.
- Carlyle, Ralph, "The Tomorrow Organization," Datamation, (Feb 1, 1990), pp. 22-29.
- Charan, Ram, "How Networks Reshape Organizations--For Results," Harvard Business Review (Sep/Oct 1991), pp. 104-115.
- Crowston, Kevin, Malone, Thomas W. and Lin, Felix, "Cognitive Science and Organizational Design: A Case Study of Computer Conferencing," Human-Computer Interaction, 3 (87-8), 59-85.

- Donovan, John J., "Beyond Chief Information Officer to Network Manager," Harvard Business Review (Sep/Oct 1988), pp. 134-140.
- Drucker, Peter F., "The Coming of the New Organization," Harvard Business Review, (Jan/Feb 1988), pp. 45-53.
- Goodstein, Leonard D. and Burke, W. Warner, "Creating Successful Organization Change," Organizational Dynamics (Spring 1991), pp. 5-17.
- Gorry, G. Anthony and Scott-Morton, Michael S., "Framework for Management Information Systems," Sloan Management Review (Spring 1989), classic reprint series, pp. 49-61.
- Hoffman, Gerald, "Coping with Information Megalomania," Information Management FORUM, AMA, (Sep 1991).
- Jones, Louise H. and Kydd, Christine T., "An Information Processing Framework for Understanding Success and Failure of MIS Development Methodologies," Information and Management, 15 (1988), pp. 49-61.
- Lucas, Henry C. Information Systems Concepts for Management. Fourth Edition. New York: McGraw-Hill, Inc., 1990, p. 12.
- Markus, M. Lynne, "Power, Politics, and MIS Implementation," Communications of the ACM, 26 (6) (1983), pp. 430-444.
- McFarlan, F. Warren and McKenney, James L., Corporate Information Systems Management The Issues Facing Senior Executives, Homewood, IL: Richard D. Irwin, Inc., 1983.
- Nolan, R., "Managing the Advanced Stages of Computer Technology: Key Research Issues," In F. Warren McFarlan, ed., The Information Systems Research Challenge. Boston, MA: Harvard Business School Press.
- Provan, Keith G., "Environment, Department Power, and Strategic Decision Making in Organizations: A Proposed Integration," Journal of Management, 15 No. 1, (1989), pp. 21-34.
- Rockart, J. F., "Chief Executives Define Their Own Data Needs," Harvard Business Review, (March/April 1979).
- Rockart, J. F. and Crescenzi, Adam D., "Engaging Top Management in Information Technology," Sloan Management Review, Summer 1984, pp. 3-16.
- Tjosvold, Dean and Chia, Lai Cheng, "Conflict Between Managers and Workers: The Role of Cooperation and Competition," The Journal of Social Psychology, 129(2), pp. 235-247.
- Zachman, John, Article in the IBM Systems Journal (Vol. 26, No. 3, 1987).



Source: Nolan, R. 1984. "Managing the Advanced Stages of Computer Technology: Key Research Issues".

# Managing the End User: An End User Computing Course for the 90s

Irv Englander, Ph.D.  
Computer Information Systems Department  
Bentley College  
175 Forest Street, Waltham, MA 02154  
e-mail: ienglander@bentley.edu

## Abstract

The End User Computing course described in this paper is an advanced elective focusing on the management and control of the end user environment, and of end user computing. The course addresses many of the concerns and issues of current importance in organizations. It uses a pedagogical model that considers end user computing from three different perspectives: that of end user, that of manager, and that of end user support services. Four different courseware components provide a variety of knowledge and experience to the student.

## Introduction

End User Computing is an advanced elective in the Computer Information Systems Department at Bentley College, offered at the undergraduate level to CIS majors and nonmajors and at the graduate level to MBA and other business majors. Despite its published reputation amongst students at the college as a course with an extremely heavy workload, it is the most popular elective offered by the department.

Unlike many textbooks with the same title but very different intent, this course does not focus on the use of common computing tools. Indeed, a course sequence introducing computer tools, the system development life cycle, and the use of information in an organization is required as a prerequisite. Instead, the course addresses a broad range of end user concerns, such as the selection and development of end user applications, the management of end user computing, and the organizational services and systems needed to provide effective support for end user computing within the organization. It introduces current problems and issues and

methods. A variety of pedagogical tools are employed.

The course is unusual in the number of topics covered, and the different points of view represented. Material is presented from the perspectives of end user, direct management, end user support personnel, and company management. A variety of different topical materials and projects serve to broaden the course beyond the textbook and allow the student to *experience* end user computing

## Rationale

There has been a phenomenal increase in the availability and use of computers by management level and professional employees in the short eleven years that have passed since the IBM PC was introduced in 1982. A study conducted in 1983 [Datamation, 1983 ] showed that between 20 and 35% of middle and senior managers and other professionals in more than 1000 corporations and institutions surveyed were using PCs, and that they were using their PCs an average of 11 hours a week. A similar study conducted in 1990 [Ryan, 1990 ] indicates

that more than 78% are presently using computers, and that 32% of those are using their computers 5 hours or more every workday. Advances in networking and distributed processing are accelerating the process.

Not surprisingly, this growth has been accompanied by numerous changes in the way computers are used, in the way corporations manage and control computer use, and in the way end users spend their time and perform their tasks. Innovative software has improved the ability of professionals to access and manipulate information, and shortens the development process, but at the same time requires more support from computer professionals in the roles of consulting, training, and basic system support. Whereas most mainframe computer operation was centrally analyzed, designed, programmed, and supported, most end users are significantly on their own, depending on computer personnel mostly for consulting, training, and basic support [Panko, 1988].

There is evidence that these factors have also resulted in problems: high costs, poor or ineffective utilization of the resources, and work inefficiency for the employee. A recent article in Infoworld indicates that an increasing number of organizations are dissatisfied with the costs of operating information centers [Currid, 1992]; conversely, users have also expressed dissatisfaction with the support available. [van Kirk, 1992]. Van Kirk suggests the need for the corporate culture to pay more attention to user issues.

The course that we have developed attempts to meet these needs. It addresses the needs of end users, discusses the problems of end user support and management, and looks at some of the new methodologies and approaches that are being used to solve some of these problems. These are issues and topics that do not receive much consideration in the traditional IS

curriculum, because they focus outside of traditional IS professional activities, yet they are of growing importance within business organizations, and become a desirable addition to the curriculum.

### Pedagogical Model

The model for the course is based on the observation that in order for end user computing to be useful and effective, the end user must be able to understand and control his/her own computing environment, to exploit its possibilities, and to understand its limitations. It is common to define the environment in terms of technology. Certainly the end user computing environment includes the technology, but ultimately the success of end user computing depends more upon the management and support environments than upon the technology itself. For that reason, the material in the course is presented from three different perspectives:

1. First, it is necessary that the end user understands the technology of his/her environment, the management of that technology, and the services that are available in support of his/her efforts. This material is presented from the perspective of the end user. Many of the tools for this course are designed to put the student into the position of end user, to allow the student to consider and appreciate the implications of that position.

2. Next, it is recognized that the management of end user computing puts special requirements and limitations on the user. Group projects must serve the needs of multiple users; budgets are limited; the cost and time for application development make some projects impractical; system standardization across an organization may bring organizational benefits at the expense of an individual user. Thus, the second perspective considered is that of management of end user computing, at the individual,

group, and organizational levels. For this purpose the student is asked to view end user computing from the perspective of management needs. In addition to the course readings, a major team project, described in the next section, addresses some of these issues.

3. The third perspective presented is that of the support system that the end user is dependent on to solve problems and to address concerns. Various forms of support are considered. Students get to see the problems and difficulties inherent in the position of end user support, and to discuss possible solutions. CIS students in the class get to understand the requirements of an end user IS professional. All the students in the class learn to appreciate the difficulties of providing effective support solutions, and the management of support services. An interesting side issue is that the graduate class usually has one or two Information Center consultants in the class. These students add a dimension to this part of the course.

Each component of the course focuses on one or more of these elements. Most focus on several.

### **Courseware Components**

In addition to the usual lectures and exams, there are four major components to the courseware. Each component is designed to reflect one or more of the perspectives presented in the model:

#### ***1. Textbook and supplementary reading.***

As a textbook, we use Panko's excellent, though dated, text, End User Computing. [Panko, 1988] The textbook material is supplemented by various readings. The supplementary readings provide current updates to the material in the textbook, add new perspectives, address current problems, and

discuss new and innovative approaches to end user computing. These additional readings are required due to the rapid pace of change and development in the field.

The topics presented are loosely based on the textbook. The lecture material and readings present the material from all three perspectives. The material introduces the technology of end user computing, discusses the issues of end user system development, expounds upon the organization, management, and requirements for end user support services, and discusses the management of end users in the context of end user computing needs. An outline of the specific topics covered is shown in table 1.

Most of the lecture time is devoted to the reading topics. One class session is devoted to the introduction and demonstration of some of the more interesting recent end user software applications.

#### ***2. Infoworld reading***

Students are required to read the current issues of Infoworld weekly. Required reading includes the front pages, the Enterprise Computing section, product reviews and comparisons, the back pages, and other news stories that seem important to them. Students are quizzed biweekly on their reading. This component provides exposure to the field, to current events in the field, and to the jargon used within the industry. The Infoworld reading provides insight into the current issues facing end users, deepens students' understanding of the technology and the end user environment, and in addition, demonstrates graphically the difficulty faced by end user support personnel in staying current in the field.

Written course evaluations indicate that students find the Infoworld component initially difficult; but easier with each succeeding week.

One recently compared it to learning a foreign language: "Initially, nothing makes any sense, but eventually it's understandable." Students also expressed the view that they felt as though they really understood current end user technology and concerns as a result of the reading. Overall, most students thought that the experience was quite valuable.

### ***3. Individual Current Topic Projects***

In addition to the Infoworld reading, students are required to prepare a brief paper and class presentation on a current topic of their own choosing related to end user computing. The topic can be hardware, software, or concept. It can be a product, a comparison of products, a class of products, an organizational or end user management issue, or a discussion of a new or current technology that was not addressed elsewhere in the course. The critical requirement is that the topic must be of current interest in the field, with references less than six months old. The Computer Selects CD-ROM is a primary reference source for this purpose. This serves the additional benefit of exposing the students to the use of CD-ROM technology.

This independent project allows the student to explore a particular topic of interest more thoroughly, and exposes the class to topics that would otherwise not be addressed, particularly, topics of interest to the students themselves. In the context of the course model, the individual project addresses the need for the students as end users to be able to search for and understand the information they require to satisfy their computing needs.

### ***4. A Team Database Application Development Project***

Teams of three students are required to find an appropriate workgroup-sized end user database application for development, and to

conduct the interviews, develop the specifications, design the application, and build the implementation, using RBase as the development tool. Each team must also present the completed project to the class. During the presentations, the class serves in the role of management to review, evaluate and critique each project. The class instructor provides initial "training" and acts as an end user consultant on the project.

Teams are created by the instructor. The criterion is to provide as wide a range of background experience as possible for each team. Only a few of the students entering this course have any relational database exposure or experience, which has not proven to be a difficulty.

The team project exposes the students to the way in which end users approach and manage a project. Within the project students get to experience many of the problems faced by end users: the difficulties choosing an appropriate project, of understanding user requirements, of using a new and unfamiliar tool, of receiving effective support, of solving the problems that inevitably arise, and of managing the project and environment. The project also reinforces the alternative methodology used in developing a small end-user oriented system. This last issue is particularly relevant to those students who have studied the standard system development life cycle approach and have not yet experienced the opportunity to evaluate critically the methodology used in a particular development project.

RBase 3.1 was selected as the tool of choice for two reasons:

1. It has a powerful, but easy to use menu driven application development facility. This means that students without

programming experience are able to fulfill the task.

2. RBase is not used elsewhere in our courses, and is not usually known by students in the course. This fulfills the requirement that the tool be initially unfamiliar to the students.

RBase 3.1 is available in a reasonably priced package with good instruction material from Course Technology, Inc. [Harrington, 1990]

### **Conclusion:**

The curriculum model that we have adopted focuses on the ability of the end user to understand, manage, and control his/her environment, and on the management and support of end user computing at all levels of the organization. The topics addressed are appropriate to the understanding of future IS professionals and business students alike. Since it seems apparent that virtually all business people will be end users in the very near future, we feel that this course effectively addresses important needs in the 1990s IS curriculum.

### **References**

- [Currid] Currid, C., "Companies Threw Out Good Information Centers with the Bad," **Infoworld**, Vol. 14, 35, August 31, 1992, p.53
- [Datamation] ----, "Micros at Big Firms", **Data-mation**, Vol. 29, 11, November 1983, p.160
- [Harrington] Harrington, J., **R:Base 3.1: Relational Database Concepts in Practice**, Course Technology, 1991
- [Panko] Panko, R. R., **End User Computing**, Wiley, 1988
- [Ryan] Ryan, A. J., "Office PCs Seeing Increase in Use," **Computerworld**, Vol. 24, 7, February 12, 1990, p.74
- [van Kirk] van Kirk, D., "IS Staff, end-users: a Failure to Communicate", **Infoworld**, Vol. 14, 49, December 19, 1992, p. 72

### **Table 1 Outline of Course Topics**

End User Computing Environments  
End User Application Development  
Data Management on the PC  
End User Computer Hardware  
End User System Software  
Managing End User Computing  
End User Support Services  
Managing End User Support  
PC-Host connectivity and Networking  
End User Applications  
DSS and EIS

# AN EMPIRICAL STUDY OF INFORMATION SYSTEMS DOWNSIZING

Arjan Sadhwani, Department of Accounting, University of Akron, Akron, OH 44325

Bindiganavale S. Vijayaraman, Department of Management, University of Akron, Akron, OH 44325

H. V. Ramakrishna, Dept. of Information and Decision Sciences, Salisbury State University, Salisbury, MD 21801

## Abstract

Due to increased pressure from upper management to cut Information Systems (IS) cost, IS departments are downsizing mainframe-based IS applications. This study examines the approaches the companies are taking towards IS downsizing, the benefits the companies are deriving from downsizing, and the risks companies perceive in transferring applications from mainframe computers to mini and networked microcomputers. This paper reports the results of a survey of 148 information systems executives.

## Introduction

In today's increasing global competition, Information Systems (IS) downsizing has become a strategic tool for improving cost effectiveness. IS downsizing is the latest trend of the 1990s in organizations and is rapidly emerging as one of the most significant phenomenon in information resource management (Bulkeley, 1990). It represents a fundamental shift in the way organizations use computers. Downsizing is a process that involves (1) moving all or a significant portion of an organization's information system applications from complex mainframe computers onto smaller systems such as network of microcomputers and workstations, and (2) transfer of some of the basic information processing functions to end users. Although downsizing does not necessarily mean getting rid of mainframe computer, the focus is to move much of the day-to-day processing and data manipulation to a less expensive microcomputers and move applications closer to end-users, who can run their applications on PCs and satisfy some of their information needs. The result of this is a complex mix of PCs, workstations, local area networks, mini computers, and mainframe computers.

The process of downsizing will enable both IS professionals and end users to use computers more effectively, integrate resources that were formerly not linked, and enable organizations to improve the use of information to gain competitive advantage. Manipulating data on PCs gives end users greater flexibility and control.

Downsizing is emerging as an important information technology solution because of the opportunities it presents to organization. Improvement in hardware and software technology, such as microcomputers, networks, relational DBMS, off-the shelf software packages have greatly increased the availability, affordability, and usability of information technologies. For example, a small network of microcomputers doing the job of a large and expensive mainframe computers sounds very attractive to top management. Many IS managers are hardpressed to justify purchase of mainframe based systems.

## Downsizing Implementation Survey

The main reason for conducting this survey was to gain information about IS downsizing implementation issues and to identify the practices and the different approaches organizations are taking towards downsizing. In order to make our findings generalizable, we collected data from diverse industry type, ranging from medium to large size companies.

## Methodology

Data for this study were obtained by using mail survey questionnaire. A confidential questionnaire was mailed to 700 information systems executive (vice-presidents and directors of IS) selected at random in two sets. The first set consists of 205 fortune 500 companies and the second set consists of 495 companies selected from a computerized database of information systems executives. The questionnaire consisted of sections on background, implementation issues of downsizing, and perceptions and expectations of downsizing. From the companies who have implemented downsizing, data on implementation strategies, approaches used, benefits realized, and their experience on downsizing were collected. The rest of the questionnaire which included perceptions and expectations of downsizing such as: concerns, resistance, benefits and risks, and applications for downsizing was answered by all respondents. Table 1 summarizes the response rates for this survey. The second column indicates the response from the first mailing of the questionnaire. 175 companies from the first set were called by phone to remind them of the survey and request the IS executives to fill out the questionnaire. The remaining 440 companies from the second set were sent reminders along with another copy of the questionnaire requesting them to respond. The third column of Table 1 indicates the response for the reminders. The total responses were 148 out of 700 with an overall response rate of 21.1%.



## Survey Results

Tables 2-4 provides information on participating companies' revenue, industry characteristics, and their annual budget for IS. Company revenue ranged from \$185 Million to \$55 Billion with more than 39% of the sample companies having more than \$1 Billion in revenues. The companies in the sample belonged to diverse industry groups with 37% of the sample belonging to manufacturing. Annual IS budget ranged anywhere from less than \$500,000 to more than \$5 Million with more than 57% of the companies having IS budgets more than \$5 Million. Out of 148 responses, 88 companies (60%) said they are implementing downsizing and the remaining 60 (40%) have not implemented downsizing so far.

## Implementation of Downsizing

Of the 88 companies who have already begun the process of downsizing, 21 companies started the process during later half of 80's and 67 companies implemented downsizing in the 90's. When the companies were asked to identify the initiator of downsizing, 28% of the companies said upper management, 12% end-user department, 60% IS department, and the rest 5% indicated Board of Directors, President, and advance IT department.

## Downsizing Models

The difficult choice facing companies is which computer platform they should employ to achieve their downsizing goal. With networked personal computers, Unix workstations, and minicomputers to choose from, this is not an easy choice. Each of these computer environments have different strengths and weaknesses.

The PC based LANs offer the advantage of wide variety of software packages and comparatively low-cost hardware and software solution. The Unix based workstations offer improved price/performance and inherent networking capabilities. Minicomputer systems provide a highly reliable multiuser environments that includes high security for sensitive data.

The problem with choosing the "best"--cost effective, reliable, compatible--computing environment is that companies have already invested in large computers, databases, and software that they cannot afford to throw away and start all over again. So the composition of the downsized system is determined by existing hardware, software, organizational policy, in-house expertise, and applications.

Although some organizations have completely downsized all their applications to PC LANs, not all businesses can be supported solely by network-based systems. The optimal computing environment is a combination of existing large platforms and distributed networks. This could be accomplished by doing much of the processing on PCs and workstations, which in turn pass

data back to a mainframe or a minicomputer in the corporate office. In some cases, applications may be moved from a mainframe onto a minicomputer, rather than PC LAN. Each type of system running applications must be examined not only in terms of its individual technical capabilities, but also in light of organization's particular needs. Further, advances in client-server technology is providing flexible system and security controls while enhancing user productivity.

When asked about the approaches that were used by companies towards downsizing, 40% of the companies responded with client/server system of networked PCs, 36% are moving host processing to networked PCs, 7% are moving from mainframe to mini, 5% have outsourced their IS functions to consulting firms, and the rest are moving from mainframe to networked workstations. As for the strategies these companies are using for downsizing, 45% of these companies are downsizing case-by-case basis. 26% of the companies are using phased implementation, 20% are downsizing in general whereas only 2% of the companies are shifting from timeshare. About the tools these organizations are using for downsizing, 22% are using relational DBMS, 20% Front-end graphical user interface, 18% terminal emulation, 18% client/server architecture, 13% SQL front end, and 7% peer-to-peer computing.

## Benefits of Downsizing

Downsizing from mainframe and mini computers to PC LANs lets companies reduce expensive maintenance contracts, reduce the size of IS staff, and to choose among competitively priced PCs and software from many vendors thereby increasing profits and return on investment. Though the major driving force behind downsizing may be reduced costs related to elimination of (or reduction in investment in) mainframe computer and mainframe based information systems, companies may achieve added benefits. There are many other benefits of downsizing that are more important than financial gains, such as more user-friendly interface, more flexible systems, improved response, greater control, closer ties between end users and IS, improved productivity of users, more effective work, as well as the ability to better integrate information systems within businesses (Douglas, 1990). Companies can also develop applications more quickly by acquiring software packages readily available for the microcomputers. Downsizing also makes computing more responsive to the end users. By shifting applications to business units, the user has much more control over development priorities, processing, and cost. Firms also achieve more flexibility to implement new technologies by leaving the proprietary mainframe world. Lastly, downsizing allows firms to respond more quickly to new business conditions.

Table 5 lists the benefits expected from downsizing. Again the benefits are listed in descending order. User

friendly systems, improved productivity of users, faster turnaround, improved response time, and more flexible systems were listed as most beneficial. closer ties between end-users and IS staff, reduced overhead, and cost reduction were listed least beneficial. This indicates that companies downsize their IS for reasons other than cutting cost even though they expect downsizing to reduce IS cost.

Table 6 indicates level of benefit expected and benefit realized so far from downsizing. Companies were asked to rate the benefits on a scale of 1 to 5, where 1 is low and 5 is high. The responses were averaged and the benefits are listed according to the decreasing average score of benefits expected. As can be seen from Table 6, greater user access to corporate information was the most important benefit expected by the companies surveyed followed by improved productivity to end-users. Improving IS staff productivity was least benefit expected. Companies realized higher productivity of end-users due to downsizing followed by improved user support. The realized cost savings was much lower than expected. Also "benefits realized by the companies so far" was less than expected in every category of benefits. This difference may be due to optimistic expectation of downsizing.

**Risks of Downsizing**

The decision to downsize an application from mainframe to networked PCs raises many complex issues of technical and business risks. Downsizing results in increase in number of microcomputer users. This presents several hardware, software, and data control issues ranging from how access to hardware is controlled to what controls are in place to prevent unauthorized access to program and data files (Thress & Pegolo, 1991).

Despite advances in LAN technology, some concerns by corporate and IS executives have slowed downsizing at many companies. Many managers are worried about maintaining control and security in LAN environments. Increasingly, large corporations are using local area networks to run mission critical accounting applications due to downsizing. This raises the important issue of LAN security and backup as large companies integrate their LANs into enterprisewide networks. One key issue is the need to secure the network against unauthorized access. Other issues arise because users are not familiar with the traditional IS disciplines of backup, disaster prevention and planning, and documentation.

Table 7 lists the risks IS executives perceive from downsizing. Security seems to be the most important issue in downsizing followed by data integrity and disaster recovery. Fault tolerance capabilities, integration of functional applications, and data redundancy seems to be the least risky issues of downsizing. IS executives were asked to rate how difficult it is to downsize applications from mainframe to LAN based environment on a scale of 1 to 9,

1 being very difficult to 9 being not-at-all difficult. Their ratings were averaged and the average score was 4.13, indicating they perceived it to be difficult, if not very difficult.

**Downsizing Concerns**

All 148 companies in the sample responded to questions on perceptions and expectations of downsizing. Companies were asked to rate some of the concerns of downsizing in their organization. Cost of downsizing was identified as most concern followed by usability of downsized system. Loss of enterprise-wide vision of the role of IS was identified as the least concern (Table 8). When IS executives were asked to rate the resistance to downsizing from different groups in the organization on a scale of 1 to 5, 1 being extreme resistance and 5 being wide acceptance, end-users were rated at 3.6, management at 3.3, and IS personnel at 3.1. This indicates that there is more resistance to downsizing from IS personnel due to the perception that downsizing reduces the size of IS department which is true in some companies.

**Candidates for Downsizing**

The difficult choice facing managers these days is not whether they should downsize their applications, but which applications and how/when to downsize. Identifying the most appropriate applications to downsize is an important part of the process of downsizing. Despite many concerns, downsizing makes sense in many situations. However downsizing may not make sense in all organizations. For downsizing to be successful, companies must carefully consider the most appropriate applications to move to PC LANs. PC LAN environment must mature further before all systems are pulled off the mainframe. Moreover IS professionals need to become more familiar with the capabilities of PC network.

The key to identifying which applications could benefit most from downsizing is to understand the organization's motivations behind downsizing in the first place (Klein, 1991). Possible motivations include cost savings, flexibility, additional functionality, and improved user support. Although each organization's motivation will determine which applications are best suited for downsizing, matching the right system for the right application is crucial for successful downsizing. Additional influencing factors might include a firm's organizational climate, structure, propensity for risk, IS strategy, and the degree of decentralization pressure exerted by senior executives and line management. As a result, companies must carefully consider the most appropriate applications to move to PC LANs.

In general, the broader the scope of an application, which is characterized by a corporate wide requirement for shared data and functionality, multiple site locations,

significant database size, the greater the risk in downsizing. Conversely, smaller the number of users the application serves, the lesser the risk in downsizing. Applications with high risk are not appropriate for migrating from mainframe computers, while those that are less risky are good candidates for downsizing. With this strategy, applications can be gradually and successfully be downsized with minimal risk. While downsizing applications, the database size, access requirements, how timely the data should be, and the volume of data required must be taken into account. Another important factor is whether the application is functionally autonomous. The greater the need for the application to communicate, interact or connect with other systems or users, the more likely that such application is critical to the organization. Such application should remain on the mainframe computer until some of the security issues in PC LANs are resolved. Conversely, the greater the functional autonomy, such as departmental and divisional information systems, the smaller the downsizing risk.

Determining whether the PCs will store highly confidential data is critical. PCs provide less physical security and are easy to steal (especially portable computers) or log into and access confidential data (PC based operating system is less secure). Unguarded telecommunications connections are also security weaknesses. This suggests greater need for physical as well as technical security. Finally, the level of system failure that can be tolerated is another important factor. Some large organizations may have redundant hardware and fault-tolerant systems that provides disaster proof environment. Other organizations may have expensive hot-site and cold-site contracts. But in the PC LAN environment, it may be easy to replace hardware or software in the event of disaster since they use off-the-shelf hardware and software.

Table 9 rank orders applications from most appropriate to least appropriate for downsizing. Applications that require minimal interaction with the host computer, easy to migrate, simpler and less risky, not friendly on mainframe, and costly to operate and maintain on mainframe were rated as most appropriate for downsizing. Mission critical applications which involve high volume data-base updating, high security, that span organizational boundaries, and integrate with other applications were rated as least appropriate applications for downsizing. These findings support the rationale for downsizing applications discussed earlier.

#### Other Issues

Table 10 gives the percentage of increase in end-user and information systems personnel training since downsizing. Almost all companies who have downsized indicated that both end-user and IS personnel training have gone up. In-house training seems to be the most popular method of training for both end-users and IS personnel.

Computer-based training is also gaining popularity because of the availability of flexible and easy-to-use software products. Vendor-operated off-site and on-site training methods are often used by many companies to train their IS personnel more than end-users.

#### Conclusion

The survey indicated that not all companies are jumping on the Downsizing bandwagon, eventhough more and more companies have started implementing downsizing in recent years because of the availability of necessary software tools on PC LANs. Though IS department is initiating the process of downsizing in most of the organizations, upper management and end-user departments are also getting actively involved in the downsizing initiating process. Client/server approach and moving applications to PC LANs seems to be most popular strategies for downsizing among the companies surveyed. Reduction in cost does not seems to be important reason for downsizing among the companies surveyed but more user friendly system, improved productivity of users, faster turnaround are considered most important reasons for downsizing. Security seems to be the biggest issue of downsizing followed by data integrity and disaster recovery. IS executives think that critical mission applications are not a good candidate for downsizing at present time. Applications which are less risky, easy to migrate, not friendly on mainframe, and cost more to operate and maintain on mainframe should be downsized first. Many IS executives stated their policy towards downsizing is to have right application on right platform, which is also known as "Rightsizing." Many companies are developing most of their new applications on PC LANs and downsize existing applications from mainframe to PC LANs depending on their need.

#### Bibliography

- Bulkeley, William M., "PC Networks Begin to Oust Mainframe in Some Companies," Wall Street Journal, May 23, 1990, pp. 1,8.
- Douglas, David P., "Putting Large Systems On PC Networks," I/S Analyzer, Vol. 28, No.1, January 1990, pp. 1-12.
- Klein, Theodore P., "Thinking of Downsizing," CIO, Vol. 4, No. 10, July 1991, pp. 20-21.
- Thress, S. C. and Pegolo, S. R., "Auditing in a Microcomputer Environment," Internal Auditing, Vol. 7, No. 2, Fall 1991, pp. 82-87.

**TABLE 1: RESPONSE RATES**

Number of Companies	Responses to I mailing	Responses to Phone/II mailing	Total Responses
First Set--205	30 (14.6%)	25 (12.2%)	55 (26.8%)
Second Set--495	55 (11.1%)	38 ( 7.7%)	93 (18.8%)
Total--700	85 (12.1%)	63 ( 9.0%)	148 (21.1%)

**TABLE 2: PARTICIPATING COMPANY CHARACTERISTICS**

Revenue	Number of Respondents
\$25 Billion and Above	2
\$10 Billion - \$25 Billion	10
\$5 Billion - \$10 Billion	11
\$1 Billion - \$5 Billion	34
\$500 Million - \$1 Billion	16
Below \$500 Million	54
Others (not marked)	21

**TABLE 3: INDUSTRY CATEGORY OF PARTICIPATING COMPANIES**

Industry	Number	Industry	Number
Manufacturing	55	Transportation	3
Holding	18	Financial Services	2
Service	13	Real Estate	1
Utility	11	Construction	1
Bank Holding	10	Hospital	1
Retail	9	Government	1
Refinery	6	Others	9
Insurance	4	Unknown	4

**Table 4: Information System/Data Processing annual budget**

Less than \$500,000	10
\$500,000 to \$1,000,000	10
\$1,000,000 to \$3,000,000	23
\$3,000,000 to \$5,000,000	17
More than \$5,000,000	85
Other (unknown)	3

**Table 5: Benefits of Downsizing (1=Least Beneficial, 5=Most Beneficial)**

User Friendly Systems	3.98
Improved Productivity of Users	3.91
Faster Turnaround	3.83
Improved Response Time	3.70
More Flexible Systems	3.66
Cost Avoidance	3.46
Increased User Participation in System Development	3.37
Cost Reduction	3.32
Reduced Overhead	3.20
Closer Ties Between End-Users and IS staff	3.16

**Table 6: Benefits Expected Vs. Realized So Far(1=Low, 5=High)**

	Expected	Realized so far
Greater user access to corporate information	3.80	2.91
To improve productivity of end-users	3.68	2.96
To lower operating cost	3.66	2.83
To save system maintenance cost	3.62	2.74
To improve user support	3.55	2.96
Savings in system development cost	3.46	2.78
Improve productivity of IS staff	3.34	2.76

**Table 7: Risks from Downsizing (1=Least Risky, 5=Most Risky)**

Security	3.77
Data Integrity	3.71
Disaster Recovery	3.67
Incomplete Audit Trail	3.54
Back-up	3.50
Difficulty in Internal Audit	3.48
Data Redundancy	3.47
Integration of Functional Applications	3.30
Fault-Tolerance capabilities	3.16

**Table 8: Concerns of Downsizing (1=Least Concern, 5=Most Concern)**

Cost of downsizing	3.21
Usability of downsized system	3.18
Flexibility of downsized system	3.09
Senior management commitment	2.91
Loss of enterprise-wide vision of the role of IS	2.58

**Table 9: Candidates for downsizing (1=Least Appropriate, 5=Most Appropriate)**

Applications that require minimal interaction with the host computer	4.26
Applications that are easy to migrate	4.24
Applications that are simpler and less risky	4.19
Applications that are not friendly on mainframe	4.18
Applications that are costly to operate and maintain on mainframe	4.11
Applications that are highly interactive (ex.,CAD)	3.98
Applications that run too slowly on the mainframe	3.94
Applications that require end-users to retrieve, manipulate and store large amount of data to get the information they need.	3.38
Applications that supply data or services to other applications	2.98
Mission critical applications	2.76
Applications that require integration with other applications	2.49
Complex applications that span organizational boundaries	2.37
Applications that require high security	2.24
Applications with high volume data-base updating	2.24

**TABLE 10: INCREASE IN TRAINING SINCE DOWNSIZING**

	End-User	IS Personnel
In-house training	88%	80%
Vendor-operated on-site training	55%	66%
Vendor-operated off-site training	47%	75%
Computer-based training	44%	51%

# **IDST: A GUI-Based Tutoring System for Learning Data Structures**

Billy B. L. Lim\*, Jean Wang, Chris Kohlbrecker, Jamie Jason  
Applied Computer Science Department  
Illinois State University  
Normal, IL 61761.  
\*BLLIM@ILSTU.BITNET

## **Abstract**

Teaching and learning data structure materials in any computer science/information system curriculum represent some of the major challenges for instructors and students. This paper presents a graphical user interface based pedagogical tool that is easy to use to aid the process. It also discusses the classroom experience in integrating the tool in a introductory course to internal data structures.

## **1. Introduction**

One of the biggest challenges for students and instructors in any computer science/information systems curriculum is to learn and teach, respectively, data structure materials. Students are often confounded by the abstract concepts introduced to them and instructors are often handicapped by the lack of instructional tools to support the task of teaching the materials. Much of the efforts in the literature emphasize only on the approaches of teaching data structures and not on the use/development of tools for that purpose (e.g., [Levine 93]).

This paper describes IDST (Internal Data Structures Tutorial), a tutoring system that can be used to help students, and thus also instructors, having difficulties conceptualizing what actually takes place when manipulating data represented in one of the many internal data structures like stacks, queues, linked lists, binary trees, heaps, etc. With many students having difficulties grasping the class materials, the intent of IDST is to provide a tool to help students in their understanding of the fundamental concepts of data structures as utilized in computer science courses such as CS1 and CS2.

IDST provides an effective, easy-to-use tool to facilitate one's understanding of basic data structure concepts in a step-by-step manner. Its use of pull-down menus, pop-up windows, and help text make the use of the system intuitive and thus no additional training is required. As a result, IDST helps reduce the amount of time that professors, teaching assistants, and debuggers must spend assisting students with

data structures problems. The potential beneficiaries of the system would be students who are enrolled in computer science/information system courses which implement data structures, or those who have the desire to learn more about data structures. The system was developed using the object-oriented features of Borland's Turbo C++ with Turbo Vision and is available for use on a DOS-based personal computer.

The remainder of this paper is organized as follows. Section 2 gives an overview of IDST. It includes a discussion of the functionality of IDST and its user interface. Section 3 summarizes the classroom experience in integrating IDST into an introductory course in data structures. The implementation of IDST is briefly described in Section 4. Finally, Section 5 gives the summary and future work.

## **2. An Overview of IDST**

### **2.1 Functionality**

IDST is a menu-driven software application that provides the users with tutorials on basic data structures and their implementation techniques. Through the use of a graphical user interface (GUI) common to many programming integrated development environments, i.e. IDE (e.g., the IDE in Turbo Pascal), IDST allows the users to perform a self-guided tutorial at their own pace by selecting the topics in which they feel a need for improvement. IDST contains topics including basic data structures (e.g., arrays), linked list and binary tree manipulations, the implementation of queues, stacks, and heaps, and the traversals of linked

lists and binary trees in variety of orders. They include forward and backward traversals for linked lists and preorder, inorder, and postorder for binary trees. All of these are achieved through the utilization of pop-up text windows containing the appropriate instructions and graphical demonstrations with colors and special effects.

## 2.2 User Interface

The initial screen of IDST provides the user with the following menu options: Introduction, Manipulation, Traversals and Uses (see Figure 1 (all figures are given in the Appendix)). These menu options may be selected via a mouse or key combination. Each main menu option activates a pull down menu from which additional menu options may be selected (see Figures 2 through 5).

The Introduction option provides the users with the ability to peruse through text associated with basic data structures in the areas of Arrays, Linked Lists, and Binary Trees (see Figure 6, which gives the first screen of text for Arrays). The explanations given by the text, which may be scrolled with the scroll bars, help reinforce the basic concepts of the underlying data structure.

The Manipulation option enables the users to view graphical demonstrations of the manipulation, i.e. insertions and deletions, of linked lists and binary trees. To facilitate step-by-step learning of the logic involved, the display of the graphics is made to synchronize with the text explanation and the pseudocode that results in the graphical presentation. Figures 7.1 through 7.11 show the different possibilities in manipulating a linked list.

The Traversals option allows the users to step through a presentation of text and graphics explaining the traversal of linked lists or binary trees. These traversals include sequential walk-through for linked lists and pre-order, in-order, and post-order for binary trees. Like the Manipulation option, the traversals are presented in a step-by-step, synchronized manner. Figures 8.1 and 8.2 depicts an easy-to-remember means of traversing a binary tree in preorder.

The Uses option presents the users with practical implementations of data structures such as Queues, Stacks and Heaps. Examples of possible implementations include array and linked list.

## 3. Classroom Experience

IDST was introduced to a data structure class for use in the learning and teaching of data structure topics. The students were first given several chapters of IDST materials to supplement traditional classroom lectures (e.g., linked list insertions and deletions). Then, after gaining knowledge of the inner workings of IDST, each of them built a chapter that contributes to the pool of materials that may be used by others in the future. Namely, they developed materials that will be used by their peers. Many felt that when trying to teach others through the materials, they gained a better understanding of the topic themselves.

As shown in Table 1, the IDST tool received generally favorable responses from the students. (The scale is from 1 to 10, 10 being best.) For the evaluation criteria that pertain to the design goals of IDST (the ones listed in the table except OVERALL VALUE), an average score of 7.82/10 was attained.

The availability of IDST for the course worked out well in that the students were able to re-live classroom experience through IDST for the topics that they feel require another run through. This helps the understanding of the materials taught in class without the need for the instructors to redo the work. Students were also able to complete their tasks of building their own tutorials on time due to the availability of IDST outside of the laboratory (and thus they were not constrained by the limited hours of the lab).

## 4. Implementation

The "core" of IDST, i.e., the hard part that laid down the foundation work, has been implemented using Borland's Turbo C++ with Turbo Vision in an object-oriented environment [Borland 92]. Because of Borland's extensive Turbo Vision class libraries, development of the IDST desktop was relatively painless. Many of the user interface elements (e.g., pop-up windows) were reused from the class library available from Turbo Vision. Since IDST is DOS-based and can be run on any basic PC, it can be widely distributed for personal use and/or used in teaching lab. This makes IDST a feasible alternative to personal tutoring sessions.



Evaluation Criteria	Score
Ease of Use (learning existing lessons)	8.5
Ease of Use (writing new scripts)	7.9
Ease of Learning (to write scripts)	7.5
Functionality	7.1
Usefulness of Features	7.3
Consistency (user interface)	9.0
User Guide	7.5
OVERALL VALUE	7.5

Table 1: Students' Evaluation of IDST

Maintenance for IDST is made relatively simple due to the use of object-oriented approach in designing and implementing the application. It is also due to the implementation of a psuedo-LOGO interpreter that allows the utilization of customized "script" files which control the execution of the graphics displays outside of the GUI. In addition to the script files, all text presentations are ASCII files which may be easily modified to accommodate any future updates.

## 5. Summary and Future Possibilities

This paper presents IDST as an alternative to personal internal data structures tutoring. It is felt that the development of IDST is worthwhile and valuable. It provides an easy-to-use, learn-at-your own pace application that students may seek out when in need of extra instructions. Since IDST is "home-grown," it may be freely distributed to students in need of the tool.

Potential future work on IDST may include the addition of information on common sorting techniques used in conjunction with the most basic of data structures like arrays and linked lists and the porting of IDST to other platforms like Windows or OS/2.

## References

- [Borland 92] *Turbo C++ 2.0* (with Turbo Vision), Borland International, 1992.
- [Levine 93] Levine, D. B., "Dealing with Different Levels of Abstraction in a Data Structure Course," *24th*

*SIGCSE Technical Symposium  
on Computer Science  
Education, Indianapolis, IN,  
March, 1993.*

## Appendix

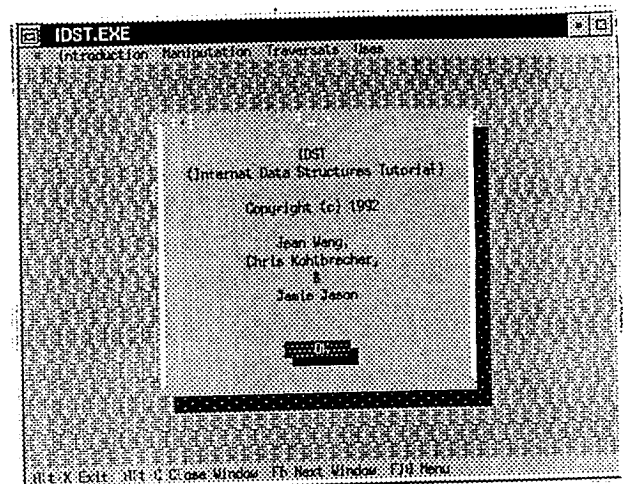


Figure 1: Initial Screen/Main Menu

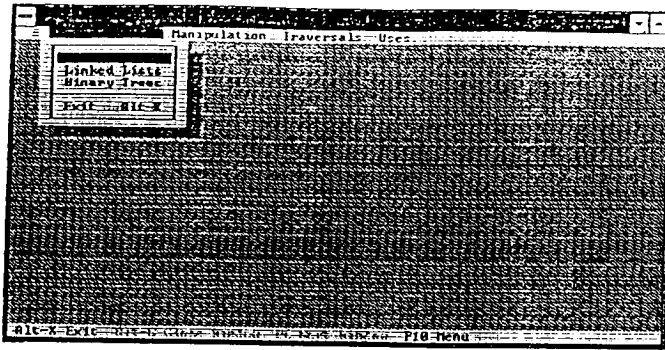


Figure 2: Introduction Pull-down Menu

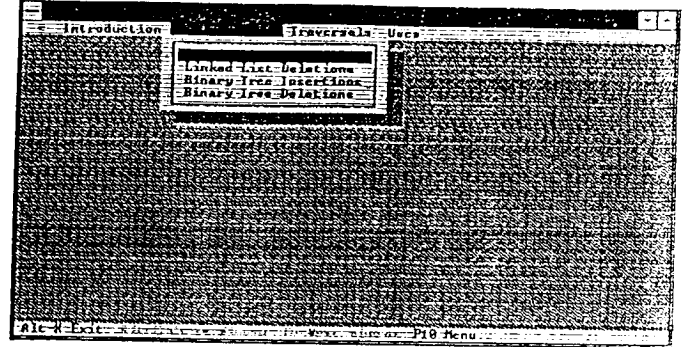


Figure 3: Manipulation Pull-down Menu

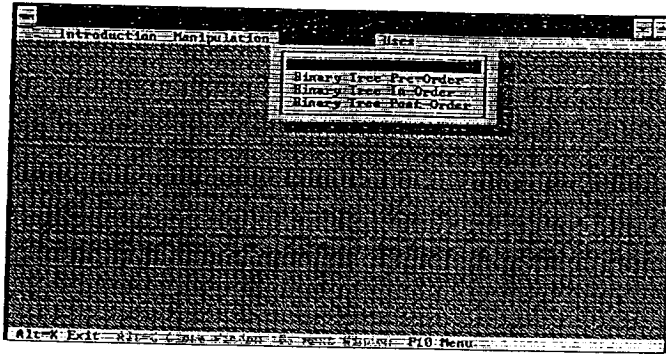


Figure 4: Traversals Pull-down Menu

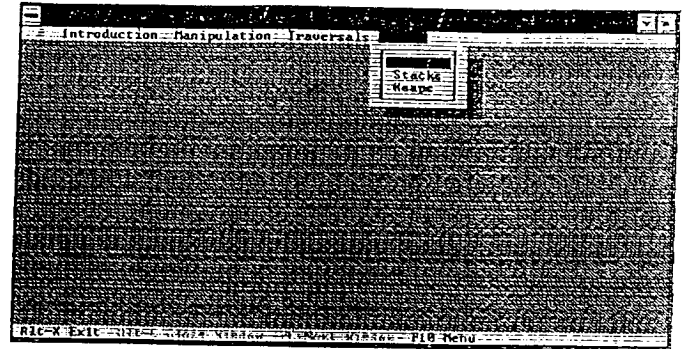


Figure 5: Uses Pull-down Menu

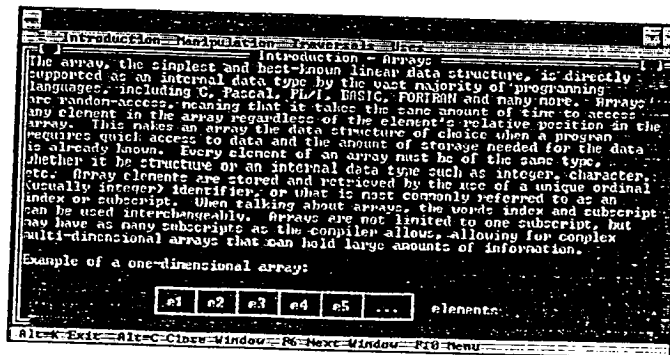


Figure 6: Arrays Menu Choice

There are three basic insertions for a linked list. The first is an insertion at the head of the list. To insert a node at the head of a linked list, follow these steps:

1. Allocate memory for the new node.

Pseudo code:

```
create new node
```

Press ENTER to continue...

There are three basic insertions for a linked list. The first is an insertion at the head of the list. To insert a node at the head of a linked list, follow these steps:

1. Allocate memory for the new node.
2. Assign the value in the header pointer to the next field of the new node.

Pseudo code:

```
create new node
newnode.next = head
```

Press ENTER to continue...

There are three basic insertions for a linked list. The first is an insertion at the head of the list. To insert a node at the head of a linked list, follow these steps:

1. Allocate memory for the new node.
2. Assign the value in the header pointer to the next field of the new node.
3. Point the header node to the newly allocated node.

Pseudo code:

```
create new node
newnode.next = head
head = newnode
```

Press ENTER to continue...

The second method is an insertion at the tail of the list. To insert a node at the tail of a linked list a pointer to the end of the linked list is needed in addition to the header pointer. To insert a node at the tail of a linked list, follow these steps:

1. Allocate memory for the new node.

Pseudo code:

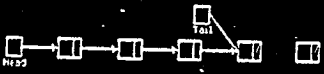
```
create new node
```

Press ENTER to continue...

Figure 7.1 - 7.4: Manipulation of Linked List

The second method is an insertion at the tail of the list. To insert a node at the tail of a linked list a pointer to the end of the linked list is needed in addition to the header pointer. To insert a node at the tail of a linked list, follow these steps:

1. Allocate memory for the new node.
2. Assign the value in the next field of the node pointed to by the tail pointer to the next field of the newly allocated node.




Pseudo code:

```
create new node
newnode.next = tail.next
```

Press ENTER to continue...

The second method is an insertion at the tail of the list. To insert a node at the tail of a linked list a pointer to the end of the linked list is needed in addition to the header pointer. To insert a node at the tail of a linked list, follow these steps:

1. Allocate memory for the new node.
2. Assign the value in the next field of the node pointed to by the tail pointer to the next field of the newly allocated node.
3. Assign the location of the newly allocated node to the next field of the node pointed to by the tail pointer.




Pseudo code:

```
create new node
newnode.next = tail.next
tail.next = newnode
```

Press ENTER to continue...

The second method is an insertion at the tail of the list. To insert a node at the tail of a linked list a pointer to the end of the linked list is needed in addition to the header pointer. To insert a node at the tail of a linked list, follow these steps:

1. Allocate memory for the new node.
2. Assign the value in the next field of the node pointed to by the tail pointer to the next field of the newly allocated node.
3. Assign the location of the newly allocated node to the next field of the node pointed to by the tail pointer.
4. Point the tail pointer to the newly allocated node.




Pseudo code:

```
create new node
newnode.next = tail.next
tail.next = newnode
tail = newnode
```

Press ENTER to continue...

The third method is an insertion in the middle of the list. To insert a node in the middle of a linked list a temporary pointer to the node to insert after is needed in addition to the header pointer. To insert a node in the middle of a linked list, follow these steps:

1. Locate the node to insert after.



Pseudo code:


```
until find node or end of list
temp = temp.next
```

Press ENTER to continue...

Figure 7.5 - 7.8: Manipulation of Linked List (cont'd)

The third method is an insertion in the middle of the list. To insert a node in the middle of a linked list a temporary pointer to the node to insert after is needed in addition to the header pointer. To insert a node in the middle of a linked list, follow these steps:

1. Locate the node to insert after.
2. Allocate memory for the new node.



Pseudo code:

```


until find node or end of list
temp = temp.next
create new node

```

Press ENTER to continue...

The third method is an insertion in the middle of the list. To insert a node in the middle of a linked list a temporary pointer to the node to insert after is needed in addition to the header pointer. To insert a node in the middle of a linked list, follow these steps:

1. Locate the node to insert after.
2. Allocate memory for the new node.
3. Assign the value in the next field of the node pointed to by the temporary pointer to the next field of the newly allocated node.



Pseudo code:

```


until find node or end of list
temp = temp.next
create new node
newnode.next = temp.next

```

Press ENTER to continue...

The third method is an insertion in the middle of the list. To insert a node in the middle of a linked list a temporary pointer to the node to insert after is needed in addition to the header pointer. To insert a node in the middle of a linked list, follow these steps:

1. Locate the node to insert after.
2. Allocate memory for the new node.
3. Assign the value in the next field of the node pointed to by the temporary pointer to the next field of the newly allocated node.
4. Assign the location of the newly allocated node to the next field of the node pointed to by the temporary pointer.



Pseudo code:

```

until find node or end of list
temp = temp.next
create new node
newnode.next = temp.next
temp.next = newnode

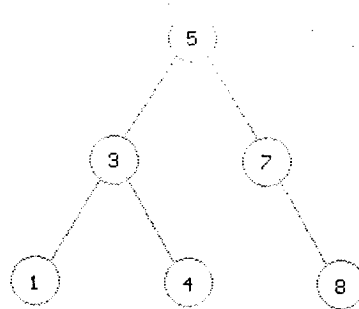
```

Press ENTER to continue...

Figure 7.9 - 7.11: Manipulation of Linked List (cont'd)

Binary Tree Transversal  
(Pre Order)

Start



ALGORITHM

```

Procedure PreOrder(R)
Begin
  If R is Empty Then
    Return
  Else
    Visit the Root R
    PreOrder(Left Subtree of R)
    PreOrder(Right Subtree of R)
  End
End
End PreOrder
  
```

Use the above aid to assist you  
in remembering the preorder tree  
transversal method:

Each node is visited as  
we pass it's left going down.

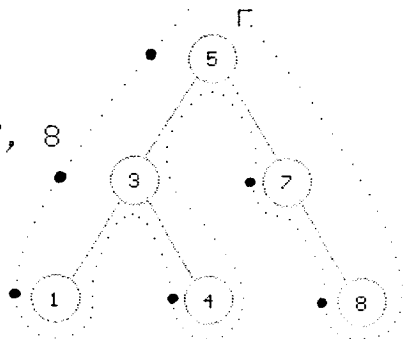
Press ENTER to continue...

Binary Tree Transversal  
(Pre Order)

Start

Result:

5, 3, 1, 4, 7, 8



ALGORITHM

```

Procedure PreOrder(R)
Begin
  If R is Empty Then
    Return
  Else
    Visit the Root R
    PreOrder(Left Subtree of R)
    PreOrder(Right Subtree of R)
  End
End
End PreOrder
  
```

Use the above aid to assist you  
in remembering the preorder tree  
transversal method:

Each node is visited as  
we pass it's left going down.

Press ENTER to continue...

Figure 8.1 - 8.2: Preorder Traversal of Binary Tree

## DESIGN COMES BEFORE PROGRAMMING

John F. Schrage, CSP, Ph.D.  
Management Information Systems Department  
Southern Illinois University at Edwardsville  
Campus Box 1106, Building II  
Edwardsville, IL 62026-1106

KEYWORD: COBOL, Program Design, Information Systems Curriculum

### ABSTRACT

To provide students with realistic work environments in programming, the traditional advanced programming course has to integrate design and implementation concepts. The programming tool used in this process can be the traditional COBOL language, but the methodology has to evolve for the design concepts. The tools for the process are becoming more automated in all hardware environments with more organizations looking at microcomputer development. Using shareware and educational software, the author has provided a worklike setting and environment to determine a prognosis becoming a programmer more than just a coder.

### INTRODUCTION

The traditional computer curriculum is based on programming and systems design classes with each taught as separate entities. Programming has evolved from the mathematical-based program to batch business applications to the screen-based interactive PC-based systems. Over the years programming became structured programming and likewise systems to structured systems. In each instance, the correctness of the end results was emphasized with productivity. The merging of the those two topics were finally noted for integration in the 1990s DPMA Model Curriculum.

In the teaching of computer programming, traditionally the instructor has taught the syntactical aspects with very minor aspects of design. The programs written have followed the list, computation, heading, control

break, table, and file types of problems. The student was in tune to get the output to look like a specific answer. From the business to technical aspects of programming, the results appeared to be the same. Almost all textbooks provided a variety of problems with the input and output specified. That concept was good in teaching the syntactical programming skills. While some topics have been added to the concepts taught, but transparency output seems to still continue.

The major negative aspect of this comes in the advanced programming course, which has been traditionally file-oriented. Advanced concepts emphasize file processing (sequential, indexed sequential, random, and occasional relative) and advanced topics on tables and data validation. In the major information systems curricula, the programming aspects have

changed to a more integrated approach. Although any language could be the tool, the investment in the COBOL language and existing code seems to provide a skewness on the actual language used. In the same manner, based on hardware resources and the PC evolution, the compiler has been changing more to a PC system than mainframe. Flexibility in student processing and design tools have been factors in this change.

The major language has somewhat waived with some programs changing, but in the back of most curricula still is the COBOL language. A change is needed in the traditional approach with the language. In an INFOWORLD editorial, five types of COBOL programmers were noted as those persons who [Metcalf]:

01. are content to only run mainframe COBOL.
02. use the PC compiler to develop and maintain mainframe COBOL.
03. have COBOL on mainframe and PC networks (cooperative processing).
04. use COBOL to move mainframe jobs to the PC network (downsizing).
05. have no mainframe but use COBOL in client/server implementations on mission-critical applications.

Discussion was noted on the object-oriented programming emphasis given to the language C++ being "little more than assembly language, roughly BCPL plus Simula67 from 25 years ago". [Metcalf] In the business articles reviewed on programming, COBOL still is the leader, even with the emphasis given to other newer languages. The Metcalfe article concludes with the following quote: "So unplug your mainframes ASAP, but don't count COBOL, BASIC, and maybe even FORTRAN out just yet".

## COURSE STRUCTURE

Over the last ten years the evolution of COBOL has changed from mainframe to include other hardware systems. The approach has emphasized more design than programming concepts. The author's 1979 experimental use of MicroFocus COBOL on an Apple II computer has progressed to the MSDOS setting with educational and full versions compilers. While microcomputers were used in a minor vein, the emphasis on the small system became evident about 1986.

The advanced course changed to the title of "File Techniques and Program Design" with a description of advanced programming and design for file processing to include designing, writing, debugging and processing of COBOL programs. The description emphasized design using a programming language tool, with the systems analysis and design course added as a prerequisite with beginning COBOL programming.

The course objectives emphasized more of the design and thinking aspects. Integration of COBOL, dBASE, and other design aids were emphasized. The course objectives expected from students are as follows:

01. interpret generic verbal problem statements and synthesize specific solution demonstrations.
02. develop mainline-oriented COBOL programs for creating, sorting, updating, or deleting files/records of the major file organizations.
03. design input and output, to include test data, for use in programs written.
04. design and implement adequate tests to determine correctness of program performance, either implied or written.



05. determine the processing and file modes for given application and write appropriate code.
06. determine and utilize data base concepts for interfacing programs with files.
07. work independently with new COBOL constructs and application requirements using reference and experimentation effectively.
08. write code using various hardware and COBOL compilers.

The course topics were traditional with all topics not just discussed but utilized in at least one environment. The topics continue standard file processing of sequential, indexed, random, and relative formats. The change in COBOL area examined from the non-ANSI mode to the 1985 ANSI standard. Even the unstructured mode of COBOL is discussed in regard to maintenance programming skills. Good and bad code is discussed with examples supplied by current and former students. While the initial processing expects good data, error processing and wrong data are discussed and expected in the ending problems. The systems control features are discussed from mainframe to small systems with some emphasis on IBM JCL. This leads then into the use of program libraries, utilities, and the various file formats that have been used. The trend of interactive processing receives over twenty percent of class discussion; likewise, the interfacing between COBOL and data base systems is treated heavily. The concept of code files as tables being loaded into the system for processing is discussed with ramification for the auditor. While the mainframe is discussed with examples, implementation of concepts is completed using the MSDOS system.

## PROGRAM TOOLS

The COBOL compiler needs to change in the academic environment. Based on discussion with programs around the country, some programs are still using ANSI 74 systems and even some have ANSI 68 compilers or non-ANSI vendor compilers. The 1985 standard should be more of a rule than any other compiler. Even IBM has announced support only for the 85 compiler system after June 1994.

Various PC compilers are available for student use with educational versions from Ryan-McFarland and CA-Realia. Micro Focus is providing a discount price to their compiler for students with discounts on the full package and add-ons to the university. Other vendors have various incentives for the use of the PC compiler. An evaluation of compilers was noted in the December 1992 issue of Corporate Computing. For the author's course, a public domain COBOL compiler is used, in addition to the commercial products, to provide students with portability issues.

Tools to support the compilers are just as important as the system. All logic flows need to be automated in some fashion. McDonnell Douglas freeware of DFDdraw and SCdraw are possible candidates as so with the Flodraw shareware. Commercial packages of Flowcharting III from Patton and Patton is available for our students, based on a donation from the vendor. CASE tools are also available from such firms as McDonnell Douglas and Texas Instruments.

Tools should not end with the compiler and charting software. The compiler's text editor makes

code entry more standard and flexible plus the demonstration version of the Intelligent COBOL Editor (ICE) from Rasmussen Software, Inc. of Portland, Oregon, provides less syntactical keying errors. In the same manner, general editors have been made available for the student to examine, such as edit (under MSDOS 5.0) and the shareware program VDE. Screen design aids have been examined with an emphasis in the shareware mode. The major packages examined have included the Egull COBOL function library, Formline, and Screen designer for COBOL. Each has advantages and disadvantage which have varied with student backgrounds. To made the code look professional, a package call XREF has been used to insert line numbers and identification.

Interfacing with other languages and systems has been attempted with varied results. The insertion of SQL code has been attempted with the shareware SSQL software. SSQL is a educational version of the SQL language with the version 3.1 having positive enhancements for student code insertion.

#### COURSE FORMAT

The first part of the weekly class session provides the lecture materials and instructions needed for the next assignment. The second part deals with specific problems students have encountered in the problem solution. Questions by students are encouraged about lecture, discussion, and assignment problems. The class session does not attempt to write specific example programs but deals with theoretical concepts. The amount of lecture decreases with question

time increasing as the course progresses.

All programs are due at the close of Friday business. While the actual programming is important, documentation is given significant weight. Data and formats used in the programs are designed by the student and included in the evaluation process. The program must be working in a production sense for the problem to be acceptable. Any program without final production output receives a maximum of 50 percent of the due points. In the same manner, late programs are penalized up to 25 percent weekly.

All assignments are submitted in an envelope which is provided. All output and materials to be submitted must fit on standard size paper. The format for completed programs is as follows, with EACH AREA CLEARLY MARKED:

01. cover sheet with name, number (student and problem), and comments about the assignment, as needed.
02. chart: system diagram/hierarchy chart/logic flow diagram, if applicable.
03. input and output (screen and paper) design format, if applicable.
04. listing of test data (records) with notation on characteristics of each particular record, such as add, change, or delete.
05. first substantial computer run on problem.
06. final run of program with output.
07. diskette with first and final program and data.

The envelope is returned at next class session with diskette and grading sheet noting specific programming concerns based on logic, documentation, and coding

rules. Students are expected to keep a copy of the production run (disk and hardcopy) of each program and its grade sheet. As part of the final examination, a disk is due with all problems in sub-directories of respective programs.

#### DESIGN AND PROGRAM EXPECTATIONS

A set of general requirements is spelled out for the student to follow. Changes to the materials have been based on feedback from students taking internships and area employers, mainly those members of the information systems advisory board.

The programming assignments are based on an application area chosen by the student. Each student examines an application or situation which would be found in a normal business (not payroll, accounts receivable or payable, or general ledger) and then review at least five articles on the application area. The student then must develop a situation that would require file processing (create, update, table access, error correction, and so forth) and reports for the above application. A file then must be designed using a record length of at least 120 characters and at least eight fields, to include the key field and at least three numeric and two alphabetic fields. The design must include a field that would be used in a table program which would access an end result. All fields should be a minimum of two characters in length for minor fields. A bibliography is required with the first problem of the sources used in the file design.

After the file is design a minimum of twenty "realistic records" must be created for the file.

The student is expect to consider adding strategically designed and placed "synthetic test records" to permit easier analysis of program performance characteristics. For each program modifying the master file, that master file is dumped to the printer at initial processing and again at program completion. Each dump has a message displayed noting the beginning and ending of specific dumps, to include the notation on which file is being listed to the printer. This help makes critical records clear to the student.

Additionally, each program provides, at a minimum, a formatted report and audit report which will include title(s), headings, page numbers, current date, and report footer to denoted end of report. For those program using screen input and output, a print screen is completed on each different screen format.

All programs are written in structured COBOL with the PROCEDURE DIVISION using the DRIVER or MAINLINE routine method of program structure. Only PERFORMs and STOP RUN are permitted in that paragraph. Normally two compilers are used for the assignment. Blank lines should not be placed in the program, only comment lines. All entries in the IDENTIFICATION DIVISION after the PROGRAM-ID entry are commented. The student name and number are specified in the commented author entry. A commented remarks section are included in the first division to noted the application and procedures being used. Comments are placed strategically in the program with each paragraph in the PROCEDURE DIVISION commented as to purpose.

## PROGRAMS SPECIFICATIONS

A minimum of six sets of problems are required in the course. Depending on time elements one or two more problems are assigned to emphasize lecture materials. The first problem set is to get to know the compiler system. The student writes a set of three individual programs to print the output from the program to be designed as follows:

01. DUMP which displays on the printer the contents of the master file with notation for the beginning and end of the file.
02. PRINT which produces a formatted report of the file with selected fields.
03. AUDIT which produces a display to the printer which summarizes file activity for the auditor, either EDP or accounting type of person. The audit program prints such items as first record, last record, nth records, total add/delete/changes, total of specific field, high/low of specific field, and other items considered essential to the auditor.

As a second set, the student creates a master sequential file and an indexed file which is then closed, reopened and dumped to the printer to prove that the file exists and what the file looks like. The formatted report and file activity audit report contents are printed for the easier perusal of the users. The student CALLS the programs created in first problem set for the dump and print routines plus printing the audit report.

The third problem set does an external loading of a set of elements for a table. There should be at least five codes in the table and the table should be read into the table program for

access in a table format. An table loading example could be using the zip code to bring in the city and state of an address or using a code for payroll classification to access the actual pay rate. This will use the sequential master file created in problem two and a new output format is created to illustrate the table working. The external table file is dumped in addition to the master file.

The fourth problem set is normally the hardest based on logic and the most unrealistic based on work expectations. The sequential file update problem (a classic technique) uses the master file previously created and updates the file contents for the following minimum situations:

01. Delete existing records from master file.
02. Add new records in proper locations in the master file.
03. Change/update record's data on the master file.

At least two transactions are used to illustrate each condition one and two noted above, and four transaction records are used to demonstrate situation 3 on two specific fields as determined by the student. The only input allowed on the update transaction records is the record locator key, those fields which will be changed or altered, and the kind of update requested code. The student needs to presume (and manually assure by marking the file dumps) that the input data is correct in format and content. As an update processing audit trail for the update program (to survive beyond student testing and development stage dumps) print each old master file record that is modified before the update, followed by the transaction record used for the update

(with notation on the output as to what should happen), and the new master file record contents after the update has been processed. The updated master file is printed in a formatted report after updating. Each record that has a change is provided a written notation on the results of the change. The formatted output and audit report should be printed at the conclusion of the problem.

The fifth problem is the most realistic and uses interactive screen concepts. This is a combination of the indexed file creation of problem two and update procedures of problem four but using the indexed file techniques. Included in this solution is the editing of input and transaction data to determine correct content, logic of actions requested with appropriate audit trailing, and non-processing of bad transactions. Data is entered using a menu approach under the guise of record additions.

The indexed file update uses the data from the fourth problem but the data is not in ascending key field order. Include in the integration of new records (or modify old ones) is a minimum of five data errors which cause the records with those errors to not be processed in the production report but listed in an error report. In the same manner at least two fields are edited as completely as possible. The output for the problem continues to include dumps of files: master (before and after update) and transaction files along with audit trail and error listings. The dumps of the updated file include only the good records that were processed. Because of the relative difficulty of read-

ing dumps, student are expected to use liberal hand labeling, color-coding, highlighting, notes, underlining, arrows, and so forth to make the changes clear. If the changes are hard to understand the grade is effected.

The sixth program provides an another view of the fifth problem but using the dBASE programming instructions. A menu program using dBASE is written to create and update the file studied this term. Again, the student assumes that the data is correct. The output includes screen dumps, formatted report, audit report, and transaction log. Additional, the student uses the file created in this assignment and designs another short file to extract data from the interface of the two files. The screen and formatted report produced uses that data from both files in a decision.

The set of other problems includes such items as report writer, relative files, integration between a set of master files, COBOL verbs which have not been used in previous programs, and interfacing COBOL with another language. One popular interface has been SQL and COBOL. An academic package, the shareware package SSQL, which emulates the SQL standard, allows for inserting SQL commands in another language.

#### STUDENT PROGNOSIS

The first time the design emphasis was used in the class less than ten percent finished the class. The only good aspect was that almost all received a top grade. Over the years the drop-

out rate has been approximately fifty percent. When the main programming mode changed from mainframe to the PC, the student completion rate became two-thirds. Currently, the completion rate is at 70 percent, with the minority completion rate about 80 percent from an original figure of less than 10 percent. The package of completed programs is used as a portfolio during job interviews for employers. Most employers involved in the interview process are anxious to see the application used by the student and which software package the student preferred dealing with the programming tools.

#### COBOL REFERENCES

Integrating a library assignment of reviewing articles on the COBOL language has create a positive view of the language. Discussion has resulted on what COBOL was to do for business and its evolution in the business environment. A binder of articles is made available for those specific references which are further examined by the class dealing with program constraints as expectations. Each student finds ten articles dealing with COBOL and reads five of the articles. The sources are broken down with at least five sources less than five years old; three sources between five and ten years old; and one source must be over ten years old. A two page review on the five articles is completed noting the complete bibliographic source for each article.

#### CONCLUSION

COBOL today should not be the COBOL of the last decade. COBOL is not dead and will not die like the punched card based on the integration of the language in the business setting. Faculty must use the programming concepts as a tool in systems analysis and design. The programming topics continue to be traditional at the onset but should evolve to screen and interactive processing.

#### Bibliography

Dodge, Marc. "Move COBOL to the Microcomputer". Corporate Computing, volume 1, number 6, December 1992, page 115.

Eliot, Lance B. and Rusty Weston. "Urgent Rescue Mission: COBOL". Corporate Computing, volume 1, number 6, December 1992, pages 72-114.

Jalics, Paul J. "COBOL on a PC: A New Perspective on a Language and its Performance". Communications of the ACM, volume 30, number 2, February 1987, pages 142-154.

Metcalf, Bob, Publisher. "Are there any COBOL users of the fifth kind? INFOWORLD, volume 14, number 47, November 23, 1992, page 39.

Snell, Ned. "Are You Ready for Cutting Edge COBOL? Datamation, October 15, 1992, pages 77-82.

# Introducing Teamwork Through Programming Courses

Luann Stemler  
Carol Chrisman

Applied Computer Science Department  
Illinois State University  
Normal, Illinois 61790-5150

Luann Stemler: (309) 438-3228      bitnet: STEMLER@ILSTU  
Carol Chrisman: (309) 438-5543      bitnet: CACHRISM@ILSTU

## Abstract

This article describes the use of collaborative activities, specifically group programming assignments, as a vehicle for introducing students to teams. The experience from two group programming assignments that were included in a programming course is reported on. The discussion includes the main advantages and lessons learned from this type of activity from both the students' and instructor's viewpoints.

## Introduction

In order to be successful in the corporate Information Systems world an employee must be a team player. They need to understand how groups operate and the roles to expect in a team. They need the skills to work effectively with other employees in collaborative situations. It is difficult for academic programs to adequately mirror this type of environment. According to Moncada [2] cooperative learning is an alternative teaching strategy that capitalizes

on group work and fosters the development of interpersonal communication effectiveness skills.

Frequently group projects are included in a few systems oriented courses such as an Analysis and Design or Database course [1,5]. Yet the feedback that Information Systems programs receive from industry consistently suggests that students need more exposure to group activities. This article will discuss introducing collaborative activities earlier in

a student's college curriculum by including a few group programming assignments.

There are a number of side benefits that relate to programming that students can gain from collaborative rather than individual programming experiences. Wilson [6] reports that collaborative work can enhance problem solving performance for novice programmers. When students work individually on programming assignments, they don't get many chances to read or debug code written by someone else which is a necessary skill for maintenance work. In programming classes students generally don't appreciate the need for planning and design before starting to write code. Students sometimes object to good design techniques like modularity and using parameters because they don't understand the need. Individual programming assignments are frequently small problems which are not representative of the complexity typically found in industrial programs. Collaborative programming assignments can address some of these issues and provide a vehicle for teaching students good design techniques.

### Activity Descriptions

In order to test the idea of introducing teamwork activities earlier than the systems oriented courses, it was decided to expose students to group activities in a programming course. A course that one of the authors regularly taught was selected where students already had experience with programming. This course required at least 2 semesters experience with a programming language. The course is targeted for transfer students from junior colleges and introduces the students to a second language. The class met twice a week for 2 hours.

As a first attempt, two group programming assignments were developed for the selected course. None of the students

had ever participated in a group programming assignment, although many of the students had participated in group projects in classes outside their computing curriculum. In each case, the assignment consisted of a single program that was large enough to benefit from modularization. The student teams had two weeks to complete the assignment. During this period, the last 20 minutes of each class was allotted for group collaboration. During this two week period, students were also working on individual programming assignments for an unrelated problem situation.

Individual teams were responsible for developing their own specific program design and determining the parameters for each module. The first week was spent on the design and programming individual modules. Each member of the team was responsible for building at least one module (sub procedure) and testing its content. During the second week the modules were put together and the team debugged the program. The team only had access to 1 copy of the team's entire program. This was possible because of the security features available on the university's mainframe computer.

### First Assignment

The first group programming assignment, assignment A, was given in the fourth week of the semester. Teams of 6 or 7 members were formed by dividing the classroom up into sections based on where the students were sitting. No attempt was made to balance the teams by ability or experience. No instructions were given on how a team should organize itself although a possible hierarchy chart was presented by the instructor to help the teams get started. The problem assigned was relatively simple so the principle challenge for the students was to determine the function of each module and the communication between modules.



None of the programs ran the first time the modules were put together. In fact none of the team programs even compiled. Typically they had problems with parameters not matching. They had failed to agree on the order and format of the parameters for each module and so individuals had made independent decisions and coded their module accordingly. Some students didn't take the assignment seriously and hadn't checked their own code. Debugging the team program was not difficult because the original problem was so simple.

The instructor had the students prepare a short paper discussing the benefits and drawbacks of group programming assignments after the first assignment. This was an attempt to understand the difficulties the students had encountered during the activity.

### Second Assignment

The second group programming assignment, Assignment B, was given in the eighth week of the semester. When asked for a preference, the students requested that the instructor again form the groups. In an attempt to balance the groups, students were arranged by current course grade and equally distributed among the groups. The second program required more complex logic to solve the problem. This time, the teams were not given any suggestions on the program design. The instructor did verbally suggest that each team appoint a leader.

Although the programs still didn't work when the individual modules were put together, the modules did fit together more smoothly. From their experience with assignment A, the teams recognized the need for some planning and so the overall design and hierarchy charts improved. However, they didn't discuss the design/logic needed

for each module. Difficulties still arose when the program was in the debugging stage. The second program required some complex mathematical logic to be applied at several stages of the program. Because of the complexity of the problem, they had logic bugs which were more difficult to debug than the simple communication mismatches of the first assignment. Students found it very difficult to debug someone else's logic. Feedback on this second assignment was obtained through a group discussion following the completion of the activity.

### Grading

Deciding how to grade group activities is always a challenge. A frequent approach in systems courses is to base the individual team member's grade on the instructor's evaluation of the team end products, peer evaluations completed by each team member, and the instructor's evaluation of each team member [1]. It is important to have the students evaluate their team experience. Spruell and LeBlanc [4] found there is an advantage to making students' assessment not only a feedback mechanism but also a learning activity.

In these group programming assignments the instructor assessed the final team program and determined a team program grade. Students evaluated themselves and the other team members, assigning individual grades on a 20 point scale. Students actually were asked to evaluate each of the modules making up the program rather than the individual students but these module grades were then interpreted as a grade for the author of the module. Individual grades for the assignment were calculated for each student by counting the team program grade equally with the average of the grades for that student from within the team.

## Student's Feedback

The students' papers discussing the group programming assignment provided feedback on the group experience and identified a number of problems which the teams encountered. Twenty-one students turned in papers after the first group programming assignment.

Lack of communication between group members was the number one obstacle reported. Fifteen students (71%) described communication problems. These communication problems arose throughout the activity. In the design stage the group members were unwilling to listen to different methods of designing a solution. During the implementation stage the group members had little regard for the technical aspects of module communication, therefore many of the programs did not compile the first time. The lack of communication made debugging increasingly more difficult.

Two secondary problems, inadequate design and lack of leadership, were described by many (42%) students. Since this was their first experience with a group programming assignment, the students did not realize the importance of the design step. They reported that the teams had difficulties deciding on the design of the program and didn't spend much effort on design. The lack of design showed when modules did more or less than what was expected by the rest of the team. There seemed to be a constant struggle to overcome the desire of each individual to write it all and the narrow focus of only being concerned with their own module.

A similar number of students (42%) felt that lack of leadership was a problem. After the fact they realized that their teams would have benefitted from choosing a leader. One student presented the analogy that there were "more chiefs than indians". This student stated that a leader would have listened to all members of the team,

organized the workloads, and helped team members to inter-relate. Because of this feedback, the instructor suggested that each team choose a leader for the second activity.

Another obstacle was the varied background experiences of the students. The first assignment came early in the semester when the students were not very well acquainted and had not taken time to assess each others skills. A third of the students thought the teams had difficulties making module assignments.

Students realized a sense of team responsibility in this assignment. Each student had the feeling of ownership toward their module. They quickly realized that when they had failed to test their own module they were letting down the entire team. Debugging was a group responsibility which not everyone accepted. Some members walked away from the debugging session stating "its not my module with the error". One student commented that the old cliché "United we stand, divided we fall" also applies to team programming activities.

Most students felt the group activity was a positive idea. Students felt this assignment would prepare them for the "real world". Students could recognize the need for better communication. One student suggested an enlarged semester long group project should be undertaken.

## Instructor's Perspective

Spruell and LeBlanc [4] claim that developing interpersonal competence along with the mastery of subject content is particularly relevant for the computer student. The instructor found the students benefitted in both of these areas from these group programming assignments.

In this class the subject content was strengthened by legitimate use of external

modules. Students were required to develop their module as an external procedure in PL/I, therefore they could not use global variables. The communication between the calling procedure and the external procedure had to be accomplished using parameters. The concept of arguments and parameter passing is one that many learners have trouble mastering. The instructor felt the first group programming assignment provided the students with a rich learning experience. In subsequent programming assignments the instructor observed more students actually using external modules.

Students also benefitted from the exposure to other people's code and to the debugging techniques of others. It is common in industry for team members to help each other debug programs but this is not always encouraged in academic settings.

Students can learn valuable lessons about how the coding style affects the readability of programs from this type of activity. Students often learn best from examples, both good and bad, and these assignments provided a rich source of examples.

There were a number of benefits in terms of the students exposure to group activities. Smart [3] found group experiences introduce students to real-world frustrations and the conflicts inherent in working relationships within groups. The group interaction in these assignments was an integral part of the learning process. Students did indeed experience frustrations typical of real group situations from these collaborative experiences.

The instructor feels that the prior group programming experience and the fact that each group had a leader both contributed significantly to the smoother running of the second assignment. With the second group programming assignment students knew better what to expect. The teams did a better job developing an overall design and planning for the communication

between modules. They left the detailed design of each module up to the individual student responsible for it. They were more willing to accept responsibility for their own module than in the first assignment. Students learned valuable lessons from this assignment on debugging. One student told the instructor he would have spent less time if he had completely rewritten the module rather than trying to fix the errors in another student's module.

### Conclusions

The use of collaborative group activities in a programming course accomplished the desired goal of introducing students to teamwork at an earlier point in their college curriculum. Since the group programming assignments occupied only a few weeks of the semester, the students gained a considerable amount of experience by investing a small amount of effort. Although they won't have the same depth of experience as if they were involved in a semester long group project, they will begin to appreciate the type of collaborative work they will encounter after graduation. A few group activities are sufficient to make students aware of many of the skills and techniques needed for a team to work effectively. In addition, the students improve their design and debugging skills through this type of group activity.

The instructor plans to make collaborative activities a standard part of this programming course. It is felt that group programming assignments would be appropriate beginning in a 2nd semester programming course. The instructor is also considering enlarging one of the group activities and having it count for the same weight as an individual programming assignment. If this is done, students will be able to concentrate their efforts on a single activity rather than trying to divide their efforts on both individual and collaborative

activities. Some thought has been given to including discussions on roles within a team and some team building activities just prior to the first group activity. The numerous benefits and the lack of significant disadvantages make collaborative assignments an idea the authors can recommend for almost all programming courses.

Conference, October 1990, Chicago, IL, pp. 7-11

6. Wilson, Judith, Nathan Hoskin, and John Nosek, "The Benefits of Collaboration for Student Programmers", SIGCSE Bulletin 25:1 (March 1993), pp. 160-164.

### References

1. Chrisman, Carol and Barbara Beccue, "Evaluating Students in Systems Development Group Projects", SIGCSE Bulletin 19:1 (February 1987), pp. 366-373.
2. Moncada, Susan, "Cooperative Learning in the IS Curriculum", ISECON '92 Proceedings of the Ninth Information Systems Education Conference, October 1992, Nashville, TN, p. 16.
3. Smart, Regina, "Introduction to Group Dynamics: A Critical Factor in System Analysis and Design Courses", ISECON '92 Proceedings of the Ninth Information Systems Education Conference, October 1992, Nashville, TN, p. 40.
4. Spruell, James and Louis LeBlanc, "A Course Planning Method to Incorporate Collaborative Learning in Information Systems Courses", Journal of Information Systems Education 4:2 (Fall 1992), pp. 6-11.
5. Tellep, Andrew, "Information Concerning Group Processes is Beneficial to Students in a Systems Development Course", ISECON '90 Proceedings of the Eighth Information Systems Education

# COBOL-85's Impact on Teaching Programming: Opportunities for Improvement

R. Wayne Headrick  
Accounting & Business Computer Systems  
New Mexico State University  
Las Cruces, New Mexico 88003  
505.646.1226 headrick@nmsu.edu

## ABSTRACT

Although COBOL-85 compilers have been available a number of years, programmers have been slow to adopt the style and teaching improvements made possible by their many added features. This paper illustrates some new techniques for solving traditional COBOL programming problems, suggests some related programming standards designed to encourage students to write more readable and maintainable code, and challenges textbook writers to embrace the full capabilities of COBOL-85.

## INTRODUCTION

The introduction of COBOL-85 made it possible to increase expectations for structured COBOL programming. Because enough time has passed since its introduction, COBOL-85 is now available virtually everywhere. Many major organizations have converted to a COBOL-85 compiler and most colleges and universities have a COBOL-85 compiler available for instructional use. Despite this, COBOL programming style has been largely unaffected. The time has come to consider a major restructuring of programming style for COBOL.

There are a number of reasons for COBOL-85's relative lack of impact. In the college and university environment, COBOL textbooks tend to be written to ensure that they are suitable for the widest possible range of audiences. This means they have generally included parallel instruction for both COBOL-85 and the older, much more limited, COBOL-74. Even the newest texts that concentrate on COBOL-85 still often include some parallel coverage of COBOL-74. Such attempts to address the COBOL-74 market put significant constraints on the programming approaches that can be advanced.

In business, there is little motivation to make major revisions in COBOL programming style to take advantage of COBOL-85's features. Because COBOL-85 was designed to be upward compatible with older versions, it has not been necessary to either learn or make use of its many, improved features to continue writing COBOL code. Of course, much of the programming being done in business is maintenance of code that have been around for years, and there is generally neither the motivation nor the

time to rewrite entire programs. Finally, and of particular interest here, as long as new programmers leaving our institutions of higher learning are being taught in the "old school", they will not be equipped to infuse new ideas and techniques into the system.

This paper suggests some new approaches to COBOL programming and challenges educators and authors to take full advantage of the features of COBOL-85. Examples are presented that demonstrate approaches to programming made possible under COBOL-85 that allow totally restructured program designs which can not be implemented under previous versions of COBOL. Because code written using the advanced features of COBOL-85 is generally improved in terms of structure, readability and style, it is time for educators and authors to abandon older versions of COBOL, completely reconsider the pedagogy, and totally rewrite example programs to take advantage of the possibilities available under COBOL-85.

## WHERE CHANGES SHOULD BE CONSIDERED

It is generally agreed that providing students with examples of good code and backing them up with well-reasoned programming standards will usually lead to improved student programs. [1] Because COBOL-85 includes many new, powerful features, it is imperative that coding examples and standards be revised together.

To demonstrate how the utilization of COBOL-85 can have a significant impact on COBOL programming style, several examples are provided below. While these examples should not be viewed as a comprehensive list of

the new features of COBOL-85, they do illustrate some of the most important and powerful features. Specifically, the EVALUATE statement (new in COBOL-85) and new features of the PERFORM and READ statements are explored in an effort provide examples that (1) focus on features of COBOL that are unique to COBOL-85, (2) illustrate code that does not have an easy equivalent in COBOL-74, and (3) show program segments that would typically be included in an introductory programming course.

### The EVALUATE statement

One of the most powerful features of COBOL-85 is the addition of the EVALUATE statement. The EVALUATE statement provides a "case" structure that is much more flexible and powerful than similar structures in other languages. [2] The EVALUATE can be used to avoid nested IFs and in most cases it allows the direct translation of selection logic from decision tables or pseudo-code into a very readable form. In its simplest form, the EVALUATE subject is a variable name that can take on one of several possible values. Figure 1 illustrates how the EVALUATE statement is used to select the appropriate action, based on SUPPLIER-CODE, from those available.

```

EVALUATE SUPPLIER-CODE
  WHEN '01'
    MOVE 'ACME SUPPLY' TO SUPPLIER-NAME
  WHEN '02'
    MOVE 'ACE SUPPLY CO.' TO SUPPLIER-NAME
  WHEN '03'
    MOVE 'BEST BUY INC.' TO SUPPLIER-NAME
  WHEN OTHER
    MOVE 'UNKNOWN' TO SUPPLIER-NAME
END-EVALUATE

```

Fig 1. Simple CASE statement using the EVALUATE

Figures 2 and 3 demonstrate a slightly different approach to the process of selecting a particular course of action from among those possible. Figure 2 shows code that will accomplish the logic for a two-level control break. Here, the EVALUATE statement correctly expresses the logic as a "case".

In Figure 3, the EVALUATE statement is used to avoid a rather messy nested IF statement. Because testing of the contents of PAY-RATE should be avoided if it is non-numeric, it is necessary to first test for that condition. Use of the EVALUATE statement in this and similar situations

```

EVALUATE TRUE
  WHEN CURR-TERRITORY-CODE IS NOT EQUAL TO
    PREV-TERRITORY-CODE
    PERFORM MINOR-BREAK
    PERFORM MAJOR-BREAK
  WHEN CURR-CUSTOMER-CODE IS NOT EQUAL TO
    PREV-CUSTOMER-CODE
    PERFORM MINOR-BREAK
END-EVALUATE
PERFORM DETAIL-PROCESSING

```

Fig 2. Control break logic using the EVALUATE

can eliminate the need for nested IFs and/or complex logical expressions. [2]

```

EVALUATE TRUE
  WHEN PAY-RATE IS NOT NUMERIC
    MOVE 'NON-NUMERIC PAY RATE' TO ERR-MSG
    PERFORM ERROR-ROUTINE
  WHEN PAY-RATE IS GREATER THAN MAX-RATE
    MOVE 'PAY RATE TOO LARGE' TO ERROR-MSG
    PERFORM ERROR-ROUTINE
  WHEN PAY-RATE IS LESS THAN MINIMUM-RATE
    MOVE 'PAY RATE TOO SMALL' TO ERR-MSG
    PERFORM ERROR-ROUTINE
  WHEN OTHER
    PERFORM PROCESS-PAY-ROUTINE
END-EVALUATE

```

Fig 3. Data validation using the EVALUATE

One of the most elegant applications of the EVALUATE statement is the direct translation of decision table logic into COBOL code. [2] The following example illustrates how the contents of a decision table can be translated directly into COBOL, avoiding any errors in the translation of decision logic and providing a very easily read implementation.

Figure 4 illustrates a simplified view of the logic necessary to process customers' orders for merchandise. Assuming that PAYMENT-INCLUDED, OVERDUE-BALANCE, and NEW-CUSTOMER are condition-names that correspond to the appropriate data-items, Figure 5 shows how the decision table logic can be directly translated into COBOL code using the EVALUATE statement. Notice that each column of the decision table relates to one of the entries in the WHEN conditions.

SITUATIONS				
Customer is new	T	F	F	
Payment is included	-	T	F	
90-day balance > 0	-	-	T	

---

ACTIONS				
Create new account	X			
Process the order	X	X		
Reject the order				X

Fig 4. Decision table logic

```

EVALUATE      NEW-CUSTOMER
              also PAYMENT-INCLUDED
              also OVERDUE-BALANCE
WHEN TRUE also ANY also ANY
  PERFORM CREATE-NEW-ACCOUNT
  PERFORM PROCESS-ORDER
WHEN FALSE also TRUE also ANY
  PERFORM PROCESS-ORDER
WHEN FALSE also FALSE also TRUE
  PERFORM REJECT-ORDER
END-EVALUATE

```

Fig 5. Logic translated into an EVALUATE

### New Features of the PERFORM statement.

A number of improvements to the PERFORM statement have been implemented in the new COBOL. One of the new features that is especially significant is the "in-line" form of the PERFORM in which the statements being PERFORMed are placed in-line rather than in an external paragraph. Previous versions of COBOL forced the use of an external paragraph to construct a loop structure even when the loop contents often consisted of only a few lines of code. Appropriate use of the in-line PERFORM can result in great improvements in readability and style since it offers the option of placing the PERFORMed code either following the PERFORM verb or in an external paragraph. [3] Another new PERFORM feature is the optional WITH TEST BEFORE/AFTER clause. As the clause indicates, the PERFORM loop can now be constructed to act either like the traditional do-while loop that has always existed in COBOL or like a do-until loop that will always be executed at least once. These new features can and should be used in an introductory programming course.

A simple but instructive example of the in-line PERFORM feature is its use in printing the contents of a two-dimensional table. Figure 6 shows the code that will print the contents of a two-dimensional summary table that has

10 rows and 5 columns onto a report in which the detail line is defined as a table consisting of a row-number and five positions for the row's column totals.

```

PRINT-TABLE.
  PERFORM PRINT-TABLE-HEADING
  PERFORM VARYING ROW FROM 1 BY 1 UNTIL ROW > 10
  MOVE ROW TO ROW-OUT
  PERFORM VARYING COL FROM 1 BY 1
    UNTIL COL > 5
  MOVE TABLE-TOTAL (ROW, COL)
    TO TOTAL-OUT (COL)
  END-PERFORM
  WRITE PRINT-RECORD FROM TABLE-DETAIL-LINE
  END-PERFORM.

```

Fig 6. Printing a table with the in-line PERFORM

Another example of the use of the in-line PERFORM is shown in Figure 7. This example demonstrates the loading of a table from a file. Again, this is a familiar task for an introductory COBOL course. Notice that the WITH TEST AFTER clause can be used here since the loop will be performed at least one time before either stopping condition will be true.

```

LOAD-RATE-TABLE.
  PERFORM WITH TEST AFTER
    VARYING TABLE-INDEX FROM 1 BY 1
      UNTIL TABLE-INDEX > 20
      OR END-OF-TABLE-FILE
  READ RATE-FILE INTO RATE-ROW (TABLE-INDEX)
  AT END
    SET END-OF-TABLE-FILE TO TRUE
  END-READ
  END-PERFORM.

```

Fig 7. Loading a table with the in-line PERFORM

A third example of the in-line PERFORM (reference Figure 8) illustrates the "get the next valid transaction record" logic often included in the introductory course. Here, the data validation routine is logically incorporated into the sequential read routine so the whole procedure can be removed from the main "process-record" routine. This approach is especially useful if data validation is to be included in a control breaks program because it makes it easy to avoid processing invalid break values.

In addition to using the WITH TEST AFTER clause of the PERFORM, this example also uses the NOT AT END clause of the READ statement discussed below. This particular paragraph is an excellent example for students to study since it elegantly demonstrates several features of COBOL-85.

```

GET-NEXT-VALID-RECORD.
  PERFORM WITH TEST AFTER
    UNTIL VALID-RECORD OR END-OF-FILE
  READ TRANSACTION-FILE INTO TRANS-WORK-AREA
  AT END
    SET END-OF-FILE TO TRUE
  NOT AT END
    SET VALID-RECORD TO TRUE
  PERFORM VALIDATE-RECORD
END-READ
END-PERFORM.

```

Fig 8. Getting the next good record with the in-line PERFORM...UNTIL

**The NOT AT END clause of the READ Statement.**

The NOT AT END clause is a new READ statement option that allows some radical changes in the structure of even the simplest COBOL programs. Under certain conditions, this clause makes it possible to eliminate the "priming" READ traditionally accomplished prior to entering a loop that processes the remaining records. The traditional approach to simple record processing, shown in Figure 9, reads the first record outside the loop and the repeated routine called PROCESS-RECORD then processes a single record and reads the next record.

```

:
PERFORM READ-A-TRANSACTION
PERFORM PROCESS-TRANSACTIONS UNTIL END-OF-FILE
:

```

Fig 9. Traditional "priming read" record processing

Figure 10 shows how the logic can be restructured to avoid use of the priming read. In this situation, the main paragraph might simply include a PERFORM PROCESS-TRANSACTIONS statement, or the code might simply be inserted into the main paragraph. Although a number of

techniques for avoiding the priming read have been developed in the past, they were usually awkward and often used the GO TO and PERFORM...THRU statements because of limitations imposed by previous versions of COBOL. While elimination of the "primary" READ is an interesting exercise, it is not readily applicable to such sequential file processing situations as the typical introductory topics of control break processing and sequential file update. As a consequence, use of this particular COBOL-85 enhancement should be approached cautiously.

```

PROCESS-TRANSACTIONS.
  PERFORM UNTIL END-OF-FILE
  READ TRANSACTION-FILE
  AT END
    SET END-OF-FILE TO TRUE
  NOT AT END
    SET TRANS-IS-VALID TO TRUE
  PERFORM VALIDATE-TRANSACTION
  IF TRANS-IS-VALID
    PERFORM PROCESS-GOOD-TRANSACTION
  END-IF
  END-READ
END-PERFORM.

```

Fig 10. Avoiding the priming read with the READ...NOT AT END

**Scope Terminators**

An explicit scope terminator has been added to each COBOL verb whose syntax includes a decision point such as READ...AT END or PERFORM...UNTIL. Much more selective than the period, the scope terminator stops only the applicable statement. As a consequence, it is now possible to write much more interesting logic than before. Because all the code presented in this paper makes use of scope terminators, the use of the external paragraph PERFORM has been greatly reduced in comparison to what would have been previously required. In fact, the only reason for the continued use of external paragraphs at all is that they often increase the readability and maintainability of the code.

**SOME STANDARDS FOR THE CLASSROOM**

Based largely on the impact of COBOL-85, the Business Computer Systems faculty at New Mexico State University has developed a set of programming standards that are used to guide students and promote the production of better



COBOL code. It is suggested that some of those standards should be incorporated into future COBOL text books.

1. NO PERFORM...THRU or GOTOs. Prior to the introduction of COBOL-85, there were some instances where programmers were virtually forced to use PERFORM...THRU and GOTO structures. It has been shown that these situations no longer exist under COBOL-85 and should be totally discouraged. [4]
2. Use only two periods in a paragraph. There are only two places in a paragraph where periods are required, one must be placed at the end of the paragraph name and the other identifies the end of the paragraph. Because COBOL-85 now has explicit scope terminators for all statements in which decision points can be included, and mixing periods with explicit scope terminators almost always leads to confusion, scope terminators should be used to the exclusion of periods inside a paragraph.
3. Eliminate use of nested IF statements. The EVALUATE statement can be used to accomplish the selection logic previously handled by nested IF's.
4. Use the in-line PERFORM...UNTIL when the number of statements in the body of the loop is relatively small. Under previous versions of COBOL, the programmer was forced to place code that was to be accomplished iteratively in a separate paragraph. This is no longer necessary and should be used only after careful consideration of the available options.

### CONCLUSIONS

A review of the COBOL examples presented in this paper should convince the reader that migration to COBOL-85 makes it possible to significantly change the structure and form of COBOL programs. In general, these changes are very positive in that they allow construction of more structured and readable code. Further, the examples show how COBOL-85 features can and should be directly applied to virtually all of the typical programming assignments used in an introductory course.

There will, of course, be disagreement and debate about some of the proposed changes. For example, the nested IF may be relatively hard for some instructors to give up. Even the author of this paper is not convinced that avoiding the priming read is a particularly good idea. The great majority of the situations in which incorporation of new COBOL-85 features can impact code development should, however, provide clearly improved results.

Authors of COBOL programming texts, both introductory and advanced, must take a leadership role in showing how to take best advantage of COBOL-85's features. To do this job properly, they will need to abandon attempts to provide parallel coverage of both COBOL-74 and COBOL-85 in the same text. The first COBOL texts to fully adopt this approach should be well received and provide bold leadership for COBOL instructors everywhere.

### REFERENCES

1. Roberts, Robert S., R. Wayne Headrick and James B. Shannon, "Using a documentation and programming standards manual in a computer information systems program," *Proceedings of the 1991 National Decision Sciences Institute Conference*, Miami Beach, FL, November 1991.
2. Headrick, R. Wayne and Robert S. Roberts, "Implementing the case statement in COBOL II," *Enterprise Systems Journal*, 6(5):86-92, May 1991.
3. Headrick, R. Wayne and Robert S. Roberts, "Improving COBOL II code with the in-line PERFORM," *Enterprise Systems Journal*, 5(11):27-30, December 1990.
4. Roberts, Robert S. and R. Wayne Headrick, "Implications of ANSI COBOL-85 on teaching: removing the last of the GOTOs," *Interfact: The Computer Education Quarterly*, 14(1):76-74, Spring 1992.

## What Directors of MBA Program Think About the Structure and Content of Information Technology Courses

B. S. Vijayaraman, Management Department, University of Akron, Akron, OH 44325

H. V. Ramakrishna, Department of Information and Decision Sciences, Salisbury State University, Salisbury, MD 21801

V. A. Quarstein, Department of Management, Old Dominion University, Norfolk, VA 23529

### ABSTRACT

This study examines the IT knowledge and skills required by MBA graduates from MBA Program Directors' perspective. An important question that is addressed in this research study is: Are the future managers, being educated/trained at business schools, adequately prepared to face the challenge posed by the expanded roles of IS in businesses operating in an increasingly competitive information-intensive global environment? This paper reports the results of a survey of 56 MBA program directors from AACSB schools of business.

### INTRODUCTION

The United States, and, indeed, the entire industrial world, has seen an extraordinary growth in MBA programs and graduates. This growth may be attributed to the quality of the product being offered; a reflection of an MBA's acquired knowledge and skills. Within the United States alone, more than 70,000 new MBA graduates left our campuses annually during the preceding 20 years according to the Graduate Management Admissions Council (GMAC, 1990). Compared to earlier growth rates of 50 percent, however, the number of MBA graduates in 1990 dropped to a mere 4.3 percent increase in the four years since 1986, when 57,000 graduated. A continuation of this decline may be attributable to a drop in the product quality. The product may no longer fit the rapidly changing organizational, social, and technological environment of the marketplace. Linden, Brennan and Lane (1992) refers to this as losing touch with reality thus causing an "MBA Glut."

Social changes that continue to affect business as reported by the GMAC are growth of technology, growing diversity of the work force for age, sex and race, and the globalization of markets, communications, and human resources. The microchip technology is recognized as the major technological change in the workplace. Use of the microchip has invaded all aspects of society as a result of lower cost and increased availability. The microchip now forms the basis for what is appropriately called "information technology." Information technology (IT) challenges managers to explore new ways of doing business (Hauge, 1987). IT poses a similar challenge to educational institutions to examine how they prepare students for their roles in society.

To what extent have MBA programs in colleges and universities reflected changes in information technology? Most MBA programs (95% of those surveyed by Chen and Willhardt, 1988) included one course in Management Information Systems (MIS). This one course meets an American Assembly of Collegiate Schools of Business requirement that has remained unchanged since 1970 (AACSB, 1990). AACSB's flexible requirements leave the questions of number, scope and contents of these courses open to deans and program directors. Such flexibility results in a variety of course offerings. Omar (1991) reports that Behling (1989) found in the latter's survey of 26 AACSB schools that 90 percent of the schools require one

course in MIS and 10 percent require two. Some programs encourage integration of MIS into other courses as a way of increasing exposure to the subject. Still others overlook MIS as a valid MBA requirement (Tyler, 1986). Despite IT's proliferation throughout businesses, time allocated to IT in MBA programs has remained unchanged since the 1970's.

Lack of definite guidelines that expand coverage of IS places MIS instructors in a dilemma. They must determine which essential facets of information technology to teach their students and which to omit. Should some aspects be taught and others dropped or should additional time be allocated to MIS? This question continually arises. Too many courses can be detrimental by taking time from other essential course requirements. Too few courses may afford insufficient coverage and exposure to a subject that has become a fundamental tool of management. Less than full exposure to IT may produce graduates who are inadequately prepared to exploit its full potential (Tyler, 1986; Carlson and Wetherbe, 1989).

### THE PROBLEM

The immediate problem of adequacy of course coverage stems from time limitations imposed on MBA course lengths and on the number of credit hours needed to graduate. The crowded MBA curriculum and the pressure to add other important courses or course coverage in such areas as international business, ethics, and the environment constrain efforts to expand coverage in the MIS area (Fowler, 1990). Some business schools have faced this problem of insufficient time to cover IT material by offering an MIS graduate degree besides the MBA degree. Insufficient coverage of IT subjects in schools represents a direct cost to businesses. That is, corporate training programs must include whatever knowledge and skills business schools omit, or managers must learn the material on their own. Because of these limitations business school deans continue to express concern that MBA's may be poorly equipped to operate in the more complex business environment being shaped by advancing microchip technology (McHenry, 1986).

Davis (1986) has identified two views that apply to the problem. First, MBA candidates need information processing as an integral part of each course, and, second, information technology must be understood before it can be

applied. The basic question is, "should business schools concentrate on further developing students with business backgrounds by increasing their technical skills or should they provide technically oriented and prepared students with an understanding of business" (Evangelauf, 1990). Business executives also reflect this dichotomy by suggesting that MBA graduates should be able to bridge the gap between business problems and technology that can help solve these problems (Carlson and Wetherbe, 1991; Simon, 1989). On the other hand, Cane and Lawrence (1990) argue that no one expects MBA's to become technical buffs, but that they should be able to understand what information systems can do and be able to evaluate proposals put forth by MIS departments. White (1990) argues that IT is the one area that management must keep up to take advantage of increasingly challenging opportunities. He further argues that, along with possessing a reasonably broad knowledge base, successful line managers should possess strong personal skills, an ability to absorb new information rapidly, and an innate grasp of internal politics.

According to Omar (1991) IT education in business schools has gone through four phases as identified by Yasin and Sawyer (1989). These are: (1) mainframe oriented courses of the 1970's, (2) microcomputer oriented courses of the 1980's, (3) microcomputer based application programs (productivity tools) such as word processing, spreadsheets and databases, and (4) use of networks to share resources. This phased sequence represents a rapid evolution of information technology and the growth curve of the industry continues to rise. The ultimate effect is an accumulation of subject material over the four phases, not a substitution of material at each phase. Thus, based on this rough reckoning, the scope of IT has approximately quadrupled in the past twenty years, but the single MIS course requirement for MBAs has remained essentially the same.

Gillenson and Stutz (1990) conducted a survey of MIS professors in AACSB masters level accredited business schools. For the MIS course in the MBA program they found that 77.3 percent of the overall sample had a course that included hardware, software, database, telecommunications, and systems analysis and design, whereas 22.7 percent did not. Of the top ten schools 75 percent said that they had such a course whereas the balance said that they had none. Beyond that, they could not define a commonly accepted approach to course design except to make the statement that three MIS subjects: MIS in the organization, MIS and corporate strategy, and systems development, were concentrated on most. The Gillenson and Stutz study could not determine what should constitute an MIS course for MBAs because guidelines had not been developed other than that promulgated by AACSB. One reason that guidelines were not available is that a properly constructed study to compare what MBA students are taught with what MBA graduates face in the workplace had not been conducted.

## RESEARCH METHODOLOGY

This paper reports results of a survey of MBA

program directors from AACSB schools of business and reflects their opinions on many of the issues raised. The program directors are key players in the design and development of MBA programs. The very nature of their work and responsibilities to manage these programs gives them a unique opportunity to understand the interaction between program structure or content and preparation of graduates for their life's work. As a key player in the dynamics of achieving educational goals and missions of a college or university, program directors are uniquely qualified to track opinions of a variety of interests such as that of recruiters, employers, faculty, and of the students themselves. One other factor suggests that program directors may represent an unbiased source of information on this particular subject area. It is the program directors' responsibility to look at the entire program without prejudice against or preference for any part or component. The MBA program directors thus represent a relatively unbiased source of accumulated knowledge and information that makes them uniquely qualified to assess the structure and makeup of the MBA program. For these reasons we obtained the opinions of program directors in AACSB schools, and attach special importance to their opinion on this subject.

### Data Collection Methods

We developed a survey questionnaire to serve as a standardized instrument for identifying changes in program makeup over time to keep track of this rapidly changing field. The rapid pace of change in IT makes it more important that assessments such as this are done more frequently. Use of responses to "actually possess" and "should possess" to most questions was required to facilitate assessment of change. Although this instrument can produce information for use in assessing program content on a one time basis, its results may also serve as a baseline for measuring changes in course structure and content in the future. After the survey instrument was satisfactorily tested, the questionnaire was sent to MBA program directors at 250 AACSB accredited schools. The questionnaire consisted of sections on background, on IT skills, on knowledge of IT concepts, and on training.

## FINDINGS AND DISCUSSION

### Sample, Participation, and Structure

The sample consisted of 56 program directors (23 percent response) who had worked at universities an average of 14.6 years (range -- 2 to 30 years). The average time in the position of programs director was 4.7 years (range -- 1 to 23 years). 37 percent held master's degrees and 63 percent doctoral degrees. 61 percent were male and 39 percent female.

As reported in other studies, not all MBA programs provide an option for MBAs to elect concentrations or to major in MIS. 58 percent of the program directors surveyed reported that they do not offer either a major or a concentration in MIS at the MBA level. 36 percent offer a concentration in MIS, and 6 percent offer a major in MIS. This latter statistic is consistent with Yaffe (1989) who found that MIS constituted 8 percent of college and university degree offerings in Los Angeles County. The

surveyed schools offered an average of five graduate MIS courses, and of these MBAs took an average of two MIS courses. The number of MBAs who elected to concentrate in or major in MIS averaged 27. These MBAs took an average of 11 additional hours of MIS subjects beyond the minimum required. Results mirror the basic questions articulated by Davis (1986) and by Evangelauf (1990) mentioned earlier regarding tradeoffs between technological knowledge and business acumen. Programs that offer one course opt for Cane and Lawrence's (1990) view that MBAs should simply be able to understand what information systems can do and be able to evaluate proposals put forth by MIS departments. 42 percent of the program directors offer programs that allow MBAs to concentrate or major in MIS thus bridging the gap between technical and business acumen as advocated by some business executives.

The immediate problem of adequacy of course coverage as a function of program content depends on the extent to which MBAs are exposed to MIS before entering the MBA programs and the extent to which MBA programs required specialized training in MIS. Concerning MIS program content, only 16 percent of the schools did not require an MIS course, whereas 77 percent required at least one course, 5 percent required two courses or more, and 2 percent required three courses. This is consistent with findings of Behling (1989) and Graf and Lauer (1992). Thus, actual practice was far from standardized and may reflect the permissiveness of the AACSB guidelines. For example, of those requiring an MIS course to graduate 57 percent required computer literacy as a prerequisite before the MBA student could take the first required course in MIS and 43 percent did not. Of those schools that did not require computer literacy, this prerequisite was satisfied in 8 percent of the cases by passing an exam, in 46 percent by undergraduate equivalent courses, or in 23 percent by taking prerequisite courses. The remaining 23 percent of the business schools assumed computer literacy, accepted non-credit course, accepted work experience, or made no formal assessment of the computer literacy question. The wide latitude permitted by AACSB guidelines also results in a wide variety of waiver standards. Waiver of the first required MIS course for MBA students was not permitted in 45 percent of the schools, permitted by proof of an undergraduate equivalent course in 30 percent, by passing an exam in 11 percent of the schools, or in 8 percent if the student possessed relevant experience, and in 7 percent if the student held an appropriate undergraduate degree. The latitude afforded by AACSB guidelines has similarly produced a variety of requirements concerning timing. MBAs were required to take the required MIS course in 48 percent of the schools during the program's first quartile, in 33 percent during the second quartile, and in 19 percent during the third quartile. This disparity further reflects the two views reported by Davis (1986). Scheduling MIS early in the program suggests that MIS is considered a prerequisite to study in other subjects, whereas, second and third quartile scheduling suggest that MIS should support only advanced subjects. Perhaps one could argue that as entering MBAs possess more and more prior exposure to IT in undergraduate courses and in their work environment, the

required MIS course can be shifted from the second or third quarter back to the first quarter, or the course itself changed from basics to more advanced subject coverage. For professional programs such as the MBA program the characteristics of the work place should determine the focus and scope of courses. If this dictum holds true, the technology that MBAs are expected to encounter in the future should have a direct bearing on course content.

#### Coding of Data

Special conventions were used to classify mean responses and to depict results in the tables for the following sections. Study results were classified according to the following procedure. The mean of responses for each topic area was categorized using the layout illustrated in Figure 1, where N refers to none or "no skill/knowledge level," B refers to beginner, I refers to intermediate and A to advanced skill or knowledge levels. Mean of the responses were computed from the survey data for the skill levels MBAs should possess (SP) and actually possess (AP).

If the mean of a response fell to the right or left of any of the three midpoints illustrated above, the skill level was assigned as N, B, I, or A, accordingly. For example, if the computed mean of responses for AP and SP were 1.7 and 2.7, a skill or knowledge level of "B" and "I" would be assigned, respectively. This would indicate that additional instructional effort and time would have to be assigned to that particular subject area to increase the skill or knowledge level of MBAs from beginner to intermediate level. If a mean fell at any of the midpoints, the lower skill level was assigned. The mean skill level difference, D, was also computed by subtracting the mean of AP from mean of SP to determine the magnitude of the change recommended.

Results as shown in Tables 1 and 2 indicate what skill and knowledge levels program directors believe MBA graduates should and actually possess. The skills are listed in the tables in rank order, within each group, from highest to lowest skill level for should possess (SP) as determined by mean responses. The skill level and order is also shown for actually possess (AP), within each group, but not in order. The skill deficiency is shown as a difference (D) in the next to last column of the table and is computed by subtracting the AP mean from the SP mean. The first column of Tables 1 and 2 indicates the overall rank order of each subject area (under should possess or SP for the entire table). Two general categories were researched: IT skills and knowledge of IT concepts.

#### Information Technology Skills

**General Skills.** Results for general skills are shown in Part A of Table 1. MBAs actually possess and should possess an intermediate skill level in the subject area of analysis of data. That is, it is not necessary to upgrade to a higher skill level, although there was still a need for improvement of skill within the intermediate level. This conclusion also applies to "generating reports." MBAs' skills were at the beginner level for "accessing external databases" and for "accessing mainframe databases," but both need to be upgraded to the intermediate level. For "downloading" and "uploading," skills were at the beginner level and should remain there, even though improvement is needed at the beginner level as indicated by the mean differences of .50

and .48, respectively.

**Hardware Skills.** Results shown in Part B of Table 1 for the skill area of "selection of hardware" suggest that MBA graduates should possess the skill level of beginner, the same as the beginner skill level they actually possess. "Installation" skill levels should remain the same at the beginner level, but for "maintenance" skills an increase from none to beginner level is indicated, even though in both instances not much effort should be required to effect these changes because the differences are .13 and .18, respectively.

**Programming Skills.** Results shown in Part C of Table 1 suggest that for programming skills program directors believed that MBA graduates should not possess skills in use of programming languages such as C Language, Fortran, Cobol, or Pascal. There is, however, some support for continuing Basic language at the beginner skill level, but the difference ( $D = .11$ ) over what they actually possessed is very small. This result represents the decline in programming in which the trend is to teach students how to use applications rather than how to program them (Fiske and Hammond, 1989).

**Software Skills.** The software skill levels MBAs should possess compared to what they actually possess are shown in Part D of Table 1. The results show that MBAs' spreadsheets skills should be increased from intermediate to advanced level. Word processing skill level should remain at the intermediate level but improvement is indicated. Statistics, graphics, and operating systems skills should continue at the intermediate level but all need considerable improvement. Database management, project management, presentation graphics, and accounting and finance skills should be increased from beginner level to intermediate level. Skill levels on use of integrated packages, desktop publishing, decision support system, and expert systems should remain at beginner level but should be improved.

**Summary.** A review of Table 1 indicates that general skills and software skills are considered about equal in importance based on a comparison of group means. These skills are followed by hardware skills, although improvement of hardware skills does not appear to be warranted except in maintenance. Programming skills are considered least important in that there is little if any requirement indicated to support their continuation in MBA programs. Table 1 also shows that all subject areas are deficient except for programming skills. Twenty-six subject areas require an increase in skill with 9 areas requiring upgrade to a higher skill level and 17 requiring an increase in skill within the same skill level.

#### **Knowledge of Information Technology Concepts**

Results presented in Table 2 indicate that program directors believe MBAs should possess much higher knowledge levels in information technology concepts than they actually possess. The four major topic areas we researched are: technology/development, applications, information resource management, and miscellaneous concepts. As shown in the following paragraphs, the differences between what MBAs should possess and what they actually possess are all positive.

**Technology/Development Concepts.** Results shown in Part

A of Table 2 indicate that MBA's knowledge of software concepts is and should remain at the intermediate level but that some improvement is needed. Database management and data communications should be increased from the beginner to intermediate knowledge levels. Application development, systems development, hardware, telecommuting, and programming concepts should remain at the beginner level, although some increase is indicated. Prototyping was now at the "none" level and should be increased to the beginner level.

**Applications Concepts.** Results shown in Part B of Table 2 indicate that MBA's knowledge of MIS, DSS, and End-User Computing concepts was at the beginner level but should be increased to the intermediate level. Office automation, distributed systems, expert systems and transaction processing were and should remain at the beginner level although there is a need to improve knowledge within that level.

**Information Resource Management (IRM) Concepts.** Results shown in Part C of Table 2 indicate that MBA's knowledge of linkages between IS strategy and business/corporate strategy was at the beginner level, but should be increased to the high end of the intermediate level. Program directors believe that use of IS for competitive advantage, data as a resource, organizational impact, the strategic role of IT, management of IS, measuring IS's effectiveness, and cost justifications of IS all warrant increases from the beginner level to the intermediate level.

**Miscellaneous Concepts.** Results shown in Part D of Table 2 indicate that MBA's knowledge of ethical issues and of legal aspects of information systems was at the beginner level and should be increased to the intermediate level. Privacy and security, disaster preparedness, ergonomics & health issues, and systems integration each was and should remain at the beginner level although the knowledge content of each of these subjects warrants increases.

**Summary.** A review of Table 2 indicates that IRM concepts top the list in terms of importance based on a comparison of group means. This is followed by Applications concepts, then closely by Miscellaneous topics, and, finally, by Technology/Development. The relative importance of subject areas within these groupings vary considerably. MBAs require additional training in all IT technology concepts subject areas, with 16 subject areas requiring an upgrade to a higher level. All except one require upgrade to the intermediate knowledge level. Fourteen subject areas require an increase in knowledge within the same knowledge area.

#### **Summary of IT Skills and Conceptual Knowledge**

The information presented in Tables 1 and 2 suggests that there are very serious shortfalls in both skill and conceptual knowledge levels of graduating MBAs. The differences between what MBAs actually possess and what they should possess measured as differences in group means indicate the magnitude of the deficiency within that group. The largest deficiencies are in Information Resources Management and Miscellaneous groups. The least deficiencies are in the Hardware Skills and Programming Skills groups. The magnitude of group means indicates the

level of importance of a group. Comparing groups in both tables shows that the most important groups are IRM concepts and General Skills followed by Software Skills. The least important groups are Hardware Skills and Programming Skills. Comparing the overall means from tables 1 and 2, IT Concepts are ranked higher than IT Skills. The need for skill and knowledge upgrades from lower to higher skill or knowledge levels is found in 25 subject areas. This condition and the need for improvement within 31 other subject areas reflects the consensus that serious course-coverage shortfalls in IT afflict many MBA programs.

### IMPLICATIONS OF THIS STUDY

The program directors' portion of the overall IT study suggests that they believe that IT constitutes an important part of an MBA's knowledge/skill base and that MBAs should possess higher skills and knowledge in almost all areas of IT when they graduate. An overall tally of the knowledge/skill levels advocated by program directors is shown in Table 3.

According to the summary of program directors opinions in Table 3, program directors believe that:

- Of the 7 no-skill areas four should remain no-skill and three should be upgraded to beginner level.
- Of the 43 subject areas at the beginner level, 22 should continue at that level of expertise and 21 should be upgraded to the intermediate level.
- Of the 7 subject areas at the intermediate level, 6 should continue at the intermediate level and one should be upgraded to the advanced level.
- Of the 57 subject areas studied, program directors recommend that 25 be taught at the beginner level, 27 at the intermediate level, and one at the advanced level.
- Of the 53 subject areas needed, 25 subject areas (marked with a double asterisk in Table 3) need to be upgraded to a higher level.
- Of the 53 subject areas needed, 28 subject areas (marked with a single asterisk) stay at the same skill or knowledge level but require an increase in coverage.

There are strong arguments based largely upon tradition that beginner, intermediate, and advanced skills and knowledge relate to one, two or three courses, respectively, at the college or university level. If this tradition can be accepted, then the 25 skills that program directors indicated should be at the beginner level means that one course should suffice to achieve this lowest level of expertise. The 27 subject areas recommended for the intermediate level would thus require a second course to achieve that rating.

Those schools of business that provide two courses in MIS in their MBA programs and those that afford students an option of concentrating in MIS have apparently already recognized this need. The balance, 58 percent of business schools according to this survey, depend upon prerequisites and other substitutes for insuring that their programs produce IT qualified MBAs. According to MBA

program directors, this is insufficient to obtain quality graduates.

Implications of this study regarding course content are also far reaching. Programming skills such as Basic and Cobol are not required according to program directors. On the other hand, advanced skills in use of spreadsheets are strongly supported. Conceptual knowledge in IRM, applications, and a variety of miscellaneous topics are considered more important than knowledge of technology and system development processes. On the skills side, software and general skills are considered very important and hardware and programming skills the least important.

The AACSB guidelines are at present too permissive to guarantee quality programs throughout AACSB accredited schools of business. The message communicated by program directors' responses in this survey clearly support a more definitive policy. Diminution of the quality of the MBA program may occur unless the AACSB upgrades requirements in information technology from one to two courses. It is clear from the statistics generated, particularly in Tables 1 and 2, that just knowing how to use a spreadsheet and word processor is inadequate preparation. Information technology includes the entire gamut of Yasin and Sawyer's four phases and is certainly not confined to just being able to use a PC (LaPlante, 1991).

This study has limitations that have been noted in previous research. First, MBAs (before and after they graduate) were declared to be key respondents in an earlier section, and their views are only indirectly included in this study. Second, this research does not include MBA program directors in schools of business that are not accredited by the AACSB.

Analysis of the data from application of the test instruments to other survey targets will help clarify and should corroborate the results of this survey of program directors. By obtaining a variety of viewpoints, answers to the questions addressed in this study may be more convincing and should alleviate some of the effect of the first limitation. Regarding the second limitation, perhaps the non-AACSB accredited category of business schools have already adjusted to the marketplace by providing MBAs with two courses to afford them intermediate level skills and knowledge in the subjects indicated. On the other hand, these schools usually use AACSB guidelines when they set up or revise their programs so this may lead to some degree of conformity.

We have had some suggestions on how to squeeze an additional course into the already crowded MBA program. We believe that a second course in IT should be offered as a substitute for another course that is less critically focused on the performance level of MBAs immediately after they graduate. First impressions are often lasting impressions. As an alternative, an additional course may be required as an add-on to the MBA program. After all, there is nothing significant about 60 credit hours except that 60 is a round number.

### REFERENCES

(References available on request)

**TABLE 1: INFORMATION TECHNOLOGY SKILLS**

	Should Possess (SP)			Actually Possess (AP)		D*
	Overall Rank**	Mean	Level	Mean	Level	
<b>A. GENERAL SKILLS</b>						
1. Analysis of Data	2	3.46	I	2.90	I	.56
2. Generating Reports	4	3.12	I	2.81	I	.33
3. Access External Databases	11	2.75	I	2.15	B	.60
4. Access Mainframe Databases	12	2.62	I	2.11	B	.51
5. Download from Mainframe	16	2.32	B	1.88	B	.50
6. Uploading to Mainframe	17	<u>2.28</u>	B	<u>1.80</u>	B	<u>.48</u>
<u>Group Means:</u>		2.76		2.28		.48
<b>B. HARDWARE SKILLS</b>						
1. Selection	15	2.34	B	1.98	B	.36
2. Installation	21	1.69	B	1.56	B	.13
3. Maintenance	23	<u>1.58</u>	B	<u>1.40</u>	N	<u>.18</u>
<u>Group Means:</u>		1.87		1.65		.22
<b>C. PROGRAMMING SKILLS</b>						
1. Basic language	22	1.62	B	1.51	B	.11
2. C language	24	1.38	N	1.24	N	.14
3. Fortran	25	1.34	N	1.32	N	.04
4. Cobol	26	1.30	N	1.34	N	-.04
5. Pascal	27	<u>1.27</u>	N	<u>1.16</u>	N	<u>.11</u>
<u>Group Means:</u>		1.38		1.31		.07
<b>D. SOFTWARE SKILLS</b>						
1. Spreadsheet	1	3.56	A	3.18	I	.38
2. Word Processing	3	3.20	I	3.00	I	.20
3. Database Management	5	3.01	I	2.48	B	.53
4. Statistics	6	3.05	I	2.61	I	.44
5. Presentation Graphics	7	2.92	I	2.34	B	.58
6. Operating system	8	2.87	I	2.67	I	.20
7. Accounting & Finance	9	2.85	I	2.50	B	.35
8. Project Management	10	2.81	I	2.15	B	.66
9. Decisions Support System	13	2.50	B	1.94	B	.56
10. DOS Shell, Windows	14	2.38	B	1.98	B	.40
11. Integrated packages	18	2.15	B	1.78	B	.37
12. Desktop publishing	19	2.13	B	1.64	B	.47
13. Expert System shells	20	<u>2.09</u>	B	<u>1.50</u>	N	<u>.59</u>
<u>Group Means:</u>		2.73		2.29		.44
<b>Overall means:</b>		2.39		2.03		.36

Note: 1. \*D refers to difference and equals the SP mean less the AP mean.  
 2. \*\* Overall rank refers to rankings (based on "Should Possess" responses) of all the 27 topics.  
 3. A score of 1 refers to "None" and 4 to "Advanced Level."

**TABLE 2 INFORMATION TECHNOLOGY CONCEPT**

	Should Possess (SP)			Actually Possess (AP)		D*
	Overall Rank**	Mean	Level	Mean	Level	
<b>A. TECHNOLOGY/DEVELOPMENT</b>						
1. Software concepts	5	2.90	I	2.56	I	.34
2. Database Management	9	2.78	I	2.18	B	.60
3. Data communication	14	2.62	I	1.93	B	.69
4. Application Development	20	2.37	B	1.90	B	.47
5. Systems Development	21	2.36	B	1.93	B	.43
6. Hardware concepts	22	2.34	B	2.11	B	.23
7. Telecommuting	27	2.23	B	1.73	B	.50
8. Prototyping	29	1.87	B	1.42	N	.43
9. Programming concepts	30	<u>1.86</u>	B	<u>1.72</u>	B	<u>.14</u>
<u>Group Means:</u>		2.37		1.94		.43
<b>B. APPLICATIONS</b>						
1. Management Information	7	2.87	I	2.47	B	.40
2. Decision Support	11	2.72	I	2.06	B	.66
3. End-User Computing	12	2.68	I	2.26	B	.42
4. Office Automation	18	2.43	B	2.02	B	.41
5. Distributed Systems	23	2.31	B	1.76	B	.55
6. Expert Systems	24	2.31	B	1.65	B	.66
7. Transaction Processing	28	<u>2.11</u>	B	<u>1.57</u>	B	<u>.54</u>
<u>Group Means:</u>		2.49		1.97		.52
<b>C. INFORMATION RESOURCE MANAGEMENT</b>						
1. Business links to IS strategy	1	3.44	I	2.31	B	1.13
2. IS for competitive advantage	2	3.14	I	2.44	B	.70
3. Data as a resource	3	3.10	I	2.42	B	.68
4. Organizational impacts	4	2.96	I	2.35	B	.61
5. Strategic role of IT	6	2.88	I	2.31	B	.57
6. Management of IS	10	2.74	I	2.39	B	.35
7. Measuring IS effectiveness	13	2.67	I	1.92	B	.75
8. Cost justification of IS	15	<u>2.56</u>	I	<u>2.07</u>	B	<u>.49</u>
<u>Group Means:</u>		2.94		2.28		.66
<b>D. MISCELLANEOUS TOPICS</b>						
1. Ethical Issues	8	2.81	I	2.04	B	.77
2. Legal aspects of IS	16	2.51	I	1.84	B	.67
3. Privacy and Security	17	2.44	B	1.86	B	.58
4. Disaster preparedness	19	2.42	B	1.70	B	.72
5. Ergonomics & health issues	25	2.30	B	1.63	B	.67
6. Systems Integration	26	<u>2.25</u>	B	<u>1.83</u>	B	<u>.42</u>
<u>Group Means:</u>		2.46		1.82		.64
<b>Overall means:</b>		2.57		2.01		.56

**Note:** 1. \* D refers to difference and equals the SP mean less the AP mean.  
 2. \*\* Overall rank refers to rankings (based on "Should Possess" responses) of all the 30 topics.  
 3. A score of 1 refers to "None" and 4 to "Advanced Level."

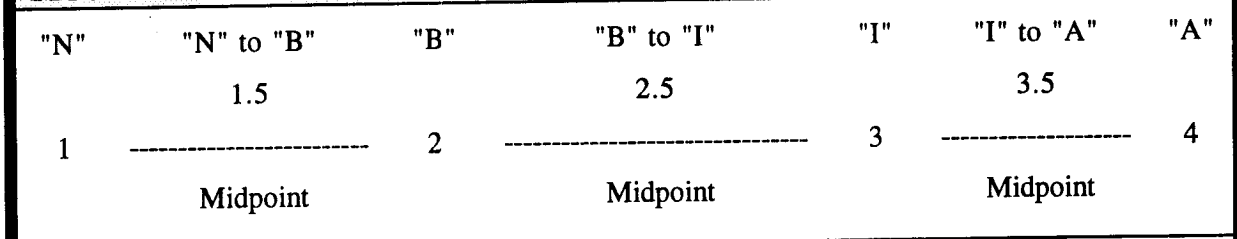


**TABLE 3: SUMMARY OF PROGRAM DIRECTORS OPINIONS**

From Actual	To Should Possess				
	None	Beginner	Intermediate	Advanced	TOTAL
None	4*	3**	0	0	7
Beginner	0	22*	21**	0	43
Intermediate	0	0	6*	1**	7
Advanced	0	0	0	0	0
<b>TOTAL</b>	4	25	27	1	57

**Note:** \* indicates subject areas stay at the same skill or knowledge level.  
 \*\* indicates subject areas need to be upgraded to a higher skill or knowledge level.

**FIGURE 1: CLASSIFICATION OF RESPONSES BY SKILL OR KNOWLEDGE LEVELS**



**N = None; B = Beginner; I = Intermediate; and A = Advanced (skill/knowledge).**

# SURVEY OF MBA-MIS PROGRAMS

James N. Morgan and Craig A. VanLengen  
both of Northern Arizona University

## Abstract

A survey of colleges and universities that offer an MIS specialization in the MBA program was conducted. Most MIS specialization MBA programs require little or no computing background of their entering students. A majority of entering students have professional job experience and about one-quarter have IS related job experience. A majority of programs reported that demand for their graduates has been increasing and that their supply of qualified applicants has been increasing. Employer and student demand are the top reasons for offering MIS specializations. The average number of hours for the MIS specialization is 13.6 with the majority devoted to technical skills. The emphasis of most of the programs is on producing IS analyst personnel.

## Introduction

This paper presents the results of a survey of colleges and universities in the United States and Canada which offer MIS specializations in their MBA programs. The survey examined: The demand for and reasons for offering an MBA-MIS specialization, the background students entering MBA-MIS programs, content of the curricula offered in these programs, and the types of jobs in which program graduates were placed.

Recent research has examined a variety of IS curriculum issues. McLoed [1985] surveyed schools to determine the content of the undergraduate MIS course, while Gupta and Seeborg [1989] conducted a similar survey at the graduate level. Chen and Willhard [1988] examined the conformance of undergraduate IS programs to the DPMA and ACM model curricula. Wagner [1990] has examined characteristics of the curricula of Masters of Science in Information Systems programs. In addition, Bialeszewski, et. al. [1989] have twice surveyed all schools with MBA programs accredited by AACSB about their IS course offerings.

Data from these surveys relating to MBA-MIS programs is extremely limited. The McLoed and Gupta surveys contain a measure of the percentage of responding schools having an IS oriented Master's Degree program. The McLoed study, conducted in 1983, found 38.9% of schools having such a program, while the 1987 Gupta survey found 51.6% of respondents with an IS oriented Masters. These estimates would include MS in IS programs as well as MBA-MIS programs. Also, the estimates are not comparable since the Gupta survey was sent only to schools having an MBA degree, while the McLoed survey was sent to some schools which had no masters degree program.

The Bialeszewski surveys are more useful for our purposes. Identical surveys were used in 1984 and 1987. These surveys focus primarily upon IS course offerings. However, they do differentiate between programs offering a doctoral degree with a major in MIS, programs offering a masters degree in IS (MSIS), programs offering an MBA with a concentration in MIS (MBA-MIS), and programs offering no specialization in MIS. Only two results relating specifically to MBA-MIS pro-

grams were reported. The percentage of responding schools having a concentration in MIS in their MBA degree was found to have increased slightly - from 45% in 1984 to 49% in 1987. The surveys also asked for estimates of the percentage of MBA students who were choosing the MIS concentration. For 1987, in 80% of the responding schools, the MBA-MIS was chosen by less than one-fifth of MBA students.

Clearly the amount of attention about MBA-MIS programs provided by these surveys is minimal. They were intended for other purposes. To gather information about MBA-MIS programs, we developed a survey questionnaire focused exclusively on the MBA-MIS program and sent that questionnaire only to schools offering an MBA-MIS specialization.

### **The Survey Instrument**

Our survey focused on four key elements of MBA-MIS programs. These are:

1. the demand for the program and reasons why a school chooses to offer an MBA-MIS specialization,
2. the background of entering students - coursework and skills requirements of entering students and work experience characteristics,
3. the content of the MBA-MIS curriculum - credit hours required, number and type of required and elective courses offered, and the topical distribution of IS coursework,
4. the distribution of placements of program graduates by type of position.

Survey forms were sent to each school listed as having an MBA program with an MIS specialization option in the 1991 Directory of Management Information

Systems Faculty [1992]. Forms were not sent to schools which have an MA or MS in a computer field but do not have an MBA with an MIS specialization. Schools having both an MA/MS and an MBA-MIS program were asked to base their survey responses only on the MBA-MIS portion of their program. The survey form was sent to the department chair or coordinator of the CS/IS department of each appropriate school in late Spring of 1992. A follow-up request was sent early in the Summer of 1992. Survey forms were sent to 111 schools. Fifty-two responses were received for a response rate of 46.9 percent. Of the 52 respondents, four indicated that they had never had, or no longer had, an MBA-MIS specialization at their school.

### **Summary of Survey Results**

Table 1 summarizes survey results with respect to characteristics of students entering MBA-MIS programs. The top portion of Table 1 looks at the level of information systems coursework or equivalent proficiency required for a student to enter the MBA-MIS program without taking remedial coursework. It is clear that most of the MBA-MIS programs are designed to serve students with minimal prior technical training. Over 60% of the respondent programs require a single introductory IS course or less of their entering students. Less than 20% of the programs require 9 or more hours of IS coursework of their entrants. Where programs do require IS coursework beyond an introductory course of their entering students, MIS courses and programming language course are the ones most frequently required.

Respondents were also asked to indicate the percentages of their entering MBA-MIS students with various work and technical backgrounds. Nearly two-thirds of the entering students have some profes-

sional job experience, while only about a quarter have IS related job experience, and just over 20 percent have undergraduate IS or CS majors in their backgrounds. The MBA-MIS program is clearly being used predominantly to build IS related technical skills in students with non-IS professional backgrounds.

**Table 1**

**CHARACTERISTICS OF ENTERING STUDENTS**

Level of IS coursework/skills required of Entering Students

	Number of Schools	Percentage of Schools
None	16	33.3%
Microcomputer Software Tools	4	8.3%
Intro. to CIS Course	11	22.9%
1 or More Prog. Languages	9	18.8%
9 or More Hours of IS Coursework	8	16.7%
	48	100.0%

Specific Courses Commonly Required of Entering Students

	Number of Schools Requiring	Percentage of Schools Requiring
MIS	9	18.8%
COBOL Programming	8	16.7%
Other Programming / Data Structures	7	14.6%
Database Management	3	6.3%
IS Anal. and Design	4	8.3%

Percent of Entering Students with:

	Ave. Across All Schools
Professional Job Experience	63.0%
IS Related Job Experience	26.0%
Undergraduate IS or CS Major	21.8%

Trends in MBA-MIS program demand and reasons for offering an MBA-MIS specialization are examined in Table 2. Demand for the MBA-MIS program can be seen in two contexts. The number of qualified students seeking enrollment in a program represents, in a direct sense, the

**Table 2**

**DEMAND FOR AND REASONS FOR OFFERING MBA-MIS PROGRAM**

Program and Student Demand Trends - Respondent's Opinion of 5 Year Trend in:

Employer Demand for Graduates

Response	Number of Schools	Percentage of Schools
Increased Sharply	5	10.9%
Increased Slightly	19	41.3%
Stayed the Same	12	26.1%
Decreased Slightly	9	19.6%
Decreased Sharply	1	2.2%

Number of Qualified Applicants

Response	Number of Schools	Percentage of Schools
Increased Sharply	4	8.7%
Increased Slightly	24	52.2%
Stayed the Same	12	26.1%
Decreased Slightly	6	13.0%
Decreased Sharply	0	0.0%

Reasons for Offering MIS Specialization in the MBA

Program Rationale	Average Rank *
Increases School's Reputation	3.125
High Student Demand for Prog.	2.667
High Employer Demand for Graduates	2.375
Program Improves Faculty Recruitment/Retention	3.354
Program Requires Minimal Additional Resources	4.063

\* Ranks used were 1 - most important to 5 - least important and 6 - not considered

demand for that program. At the same time, employer demand for graduates is an important element of the demand for a program. Our survey respondents were asked to address these issues by describing the trend over the last five years in the demand for their MBA-MIS program. They were asked to describe the trend both in employer demand for their graduates and in the number of qualified students seeking admission to their program. Overall, the results shown suggest a modest trend toward more demand for MBA-MIS programs. This was true for both the number of qualified applicants and employer demand for graduates, although the positive trend in the number of students was a bit stronger than that of the demand for graduates.

The last set of results shown in Table 2 deals with the reasons for offering an MBA-MIS specialization. Respondents were asked to rank five statements describing alternative reasons for offering graduate MBA-MIS. They were to rank the statements from 1 to 5, with 1 representing the most important reason for offering the program and 5 representing the least important. Respondents were allowed to write in their own sixth reason and to assign a rank of 6 to any statement which they felt played no role in the decision to offer the MBA-MIS program. Thus, a low average rank for one of the stated reasons for offering the MBA-MIS means that reason was considered very important.

Employer demand for graduates was seen as the most important factor, followed closely by student demand for the program. Enhancement of a school's reputation and help in recruiting and retaining faculty were also seen as important reasons for offering MBA-MIS programs. However, their importance was clearly seen as secondary to program demand factors.

**Table 3**  
**CONTENT OF MBA-MIS CURRICULUM**

Credit Hour Requirements	
Category	Average Hrs. Required
Total Hours Required for Graduation	47.2
MIS Hours Req. of All MBAs	2.1
IS Hours for MIS Specialization	13.6
Hours in Specific Req. Courses for IS Specialization	5.6
IS Elective Hours (IS hrs. offered - hrs. req.)	4.4

Topical Distribution of IS Coursework	
Topic	% of IS Work Devoted to Topic
Human Skills	16.8%
Conceptual Skills	16.8%
General Technical Skills	31.3%
Technical Design-Development Skills	35.3%

Emphasis of Program		
Emphasis	Number of Schools	Percentage of Schools
Management of IS	17	39.5%
IS Analyst	23	53.5%
Operations Research	3	7.0%
	43	100.0%

Table 3 summarizes characteristics of the MBA-MIS curricula. First, hours requirements are summarized. The average total number of hours required for graduation is 47.2. Thirteen programs require 36 hours or less, conceivable allowing graduation in one year, while 12 programs require 60 hours or more. The average amount of IS coursework required is 13.6, or about four and a half courses. This

means that just over 28 percent of the coursework in a typical MBA-MIS program is devoted to the IS specialization courses. Typically, a bit less than half of the IS hours needed for graduation, about two courses, are in the form of specific required courses. Programs vary greatly in this regard. There are 20 programs having no specified required courses and 12 programs whose entire IS content is in the form of specific required courses. The typical MBA-MIS program offers a fairly limited amount of choice for IS electives. The number of IS hours offered is on average 4.4 hours (1-1/2 courses) greater than the number of IS hours required for the specialization.

The next element of Table 3 deals with the topical distribution of IS coursework in a broad sense. Respondents were asked to estimate the proportion of IS coursework falling into each of the four broad categories, shown in Table 3. The categories used were adapted from categories of skills developed by Wagner [1990] and Klingman [1988] to describe the range of skills needed by IS professionals completing masters level programs. The following descriptions of the categories were presented to the respondents.

**Human skills:** Dealing with employees, budgeting, scheduling, planning, and communication.

**Conceptual skills:** Development of a research orientation so that students can keep up with the field after the conclusion of their formal course work.

**General technical skills:** A sufficient Knowledge of the technology to be able to communicate effectively with vendors and the MIS staff. To learn the why as well as how of various MIS tools and techniques.

**Technical design and development skills:** Detailed knowledge of systems analy-

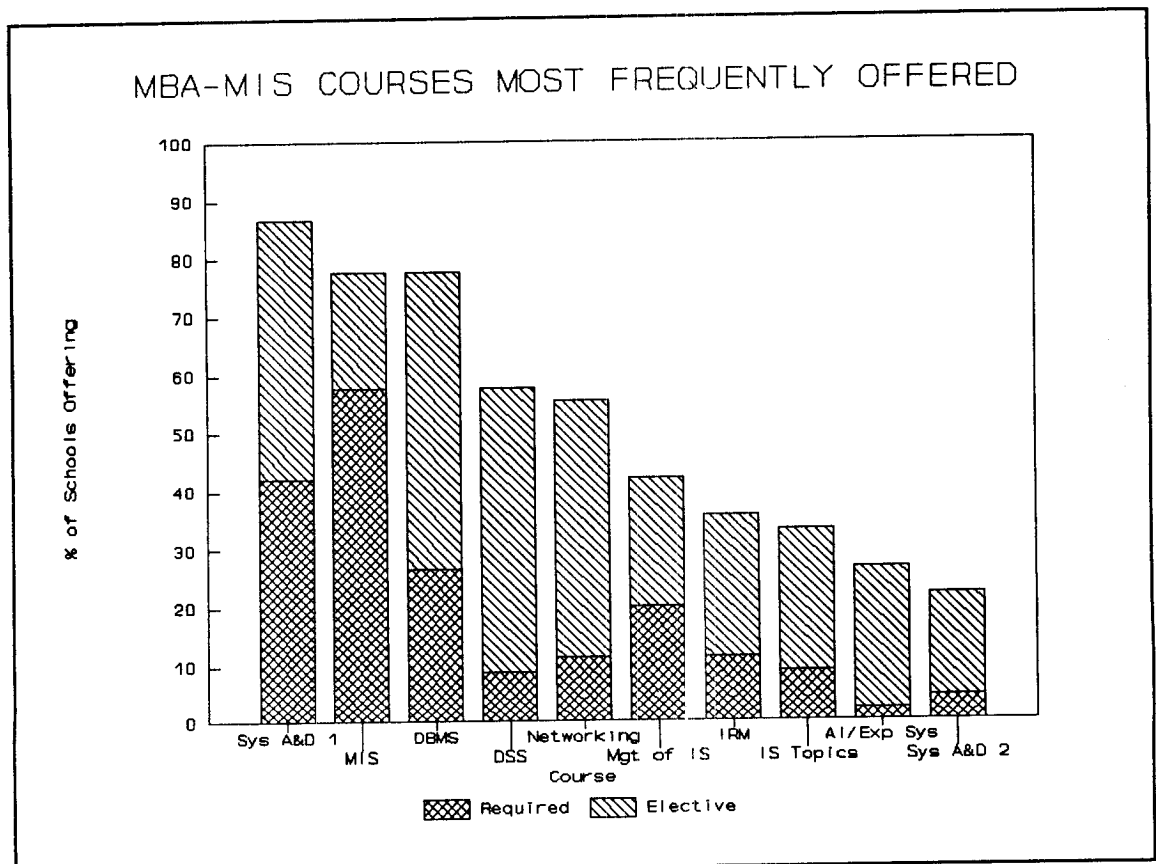
sis, design and implementation processes.

The responses suggest that the IS coursework in MBA-MIS programs centers largely on technical skills. About a third of IS coursework is normally devoted to technical design and development skills and another third is devoted to general technical skills. The remaining third is about evenly split between human skills and conceptual skills. It should be noted, however, that this response was restricted to IS coursework only. The concentration of human skills and conceptual skills building activities may be much higher in the core MBA curricula of the responding schools.

The last item in Table 3 provides a categorization of the emphasis of MBA-MIS programs. Programs were categorized based on the type of courses required and available electives. If the program required courses in systems analysis and design and database, with required or elective courses in networking and decision support systems, it was classified as emphasizing IS Analyst skills (53.5%). In some cases database and networking may have not been specifically required but the number of elective courses offered indicated that those specific courses would most likely be taken.

If the requirements and available electives did not require systems analysis and design but included at least two courses from management of information systems, management information systems, and information resource management, the program was classified as management of IS (39.5%).

The OR classification was used for those programs that required operations research courses and courses in mathematical simulation or programming (7.0%).

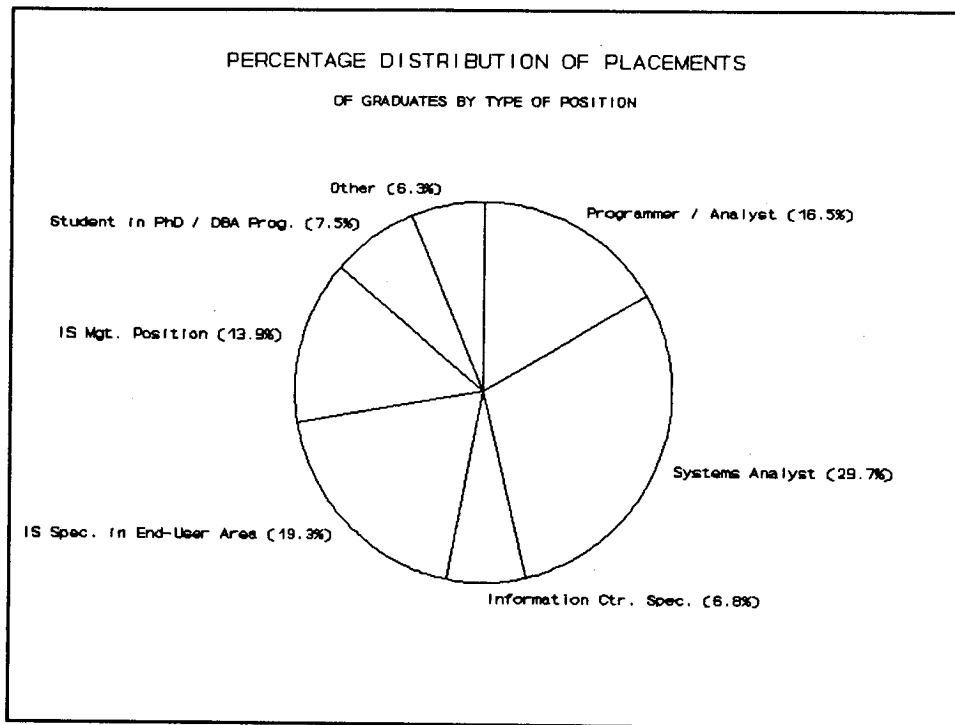


**Figure 1**

Figure 1 presents a stacked bar graph summarizing specific course offerings. The ten most commonly offered IS courses in MBA-MIS programs are shown. The cross-hatched portion of the bar for each course indicates the proportion of schools requiring the course, while the additional portion of the bar adds schools offering the course as an elective. A first course in Systems Analysis and Design, the graduate MIS course, and a course in Database Management Systems are the three most prevalent courses, being offered in about 80 percent of programs. It should be noted that the MIS course is a part of the core required of all MBAs in about half of the programs. The next most commonly offered courses are in Decision Support Systems and Networking and Telecommunications. Such courses are offered in

nearly 60 percent of programs, although they are seldom required courses. Management oriented courses in the Management of Information Systems and Information Resource Management are the next most frequently offered. About 40 percent of all MBA-MIS programs offer each of these courses. Completing the top 10 are IS Topics courses, courses in Artificial Intelligence and Expert Systems, and second courses in Systems Analysis and Design.

Figure 2 summarizes the distribution of placements of MBA-MIS program graduates across alternative categories of position. Among students not going on for further schooling, placements are about evenly split between traditional entry level IS positions and other placements. Just over 45 percent of program graduates are placed in positions as systems analysts or



**Figure 2**

programmer analysts. A bit less than 15 percent take management positions in IS. Placements as IS specialists working within end-user functional areas are quite common, nearly 20 percent, while a relatively small number of graduates, about 7 percent are taking positions as information center specialists.

### BIBLIOGRAPHY

Bialeszewski, D., Buffington, J., and Gok, M., "A Longitudinal Study of MIS Curricula at the Graduate Level," *Interface*, Vol. 10, No.1, 1988, pp. 19-23.

Chen, J. and Willhard, J., "Computer Curricula in AACSB Accredited Business Schools (1987)," *Interface*, Vol. 10, No. 1, 1988, pp. 28-31.

Gupta, J. and Seeborg, I., "The Graduate MIS Course in the Schools and Colleges

of Business," *Journal of Management Information Systems*, Vol. 5, No. 4, 1989, pp. 125-136.

Klingman, D., *CIS Profiles in Excellence Newsletter*, Vol. 3, No. 2, 1988.

McLoed, R., "The Undergraduate MIS Course in A.A.C.S.B. Schools," *Journal of Management Information Systems*, Vol. 2, No. 2, 1985, pp. 73-85.

MISRC., 1992 Directory of Management Information Systems Faculty in the United States and Canada. MISRC/McGraw-Hill.

Wagner, J., "Perspectives on the Master's in MIS", *Issues in Information Systems Education: The Proceedings of the Eighth Information Systems Education Conference, Chicago, IL, Oct. 12-14, 1990*, pp. 141-145.



# INTEGRATING IEF INTO THE CSUS UNDERGRADUATE MIS CURRICULUM\*

Thomas E. Sandman  
MIS Dept., School of Business Administration  
California State University at Sacramento  
6000 J Street, Sacramento, CA 95819-6088  
Office: (916)278-6670, Fax: (916)278-6757, E-Mail: sandmant@csus.edu

**Abstract.** *Through the generous support of Hewlett-Packard, Intel, and Texas Instruments, the Management Information Sciences Department in the School of Business Administration at CSUS has recently become a participant in the Information Engineering Facility (IEF) Academic Program offered by Texas Instruments. This program will provide an industry leading Integrated Computer Assisted Software Engineering (ICASE) lab to our students. The purpose of this paper is to describe our current intentions for integrating this technology into our undergraduate curriculum.*

## 1. Introduction.

For the last year, our MIS Department has been working to create a high technology Integrated Computer Assisted Software Engineering (ICASE) lab. These efforts will come to fruition this summer when we install a 15 workstation ICASE lab in the School of Business Administration. This lab is possible with the generous support of Hewlett-Packard, Intel, and Texas Instruments. Hewlett-Packard is donating the equipment (486 workstations, a server, and a laser printer) through their University Equipment Grant Program. Intel, whose corporate Information Engineering Facility (IEF) support center is located nearby, has become our corporate sponsor for the IEF Academic Program. Texas Instruments has been very supportive of our efforts to participate in the IEF Academic Program throughout the last year.

The purpose of this paper is to describe, now that the ICASE Lab is a reality, how we could integrate IEF into our curriculum. This discussion begins with a summary of our current undergraduate curriculum. In §3, phases of the Information Engineering (IE) Methodology are mapped to our curriculum. A new IEF related course being proposed is presented in § 4. Section 5 presents some related issues, and a summary is given in § 6.

## 2. Current Undergraduate Program.

Our current undergraduate program is summarized in Figure 1. Our department offers

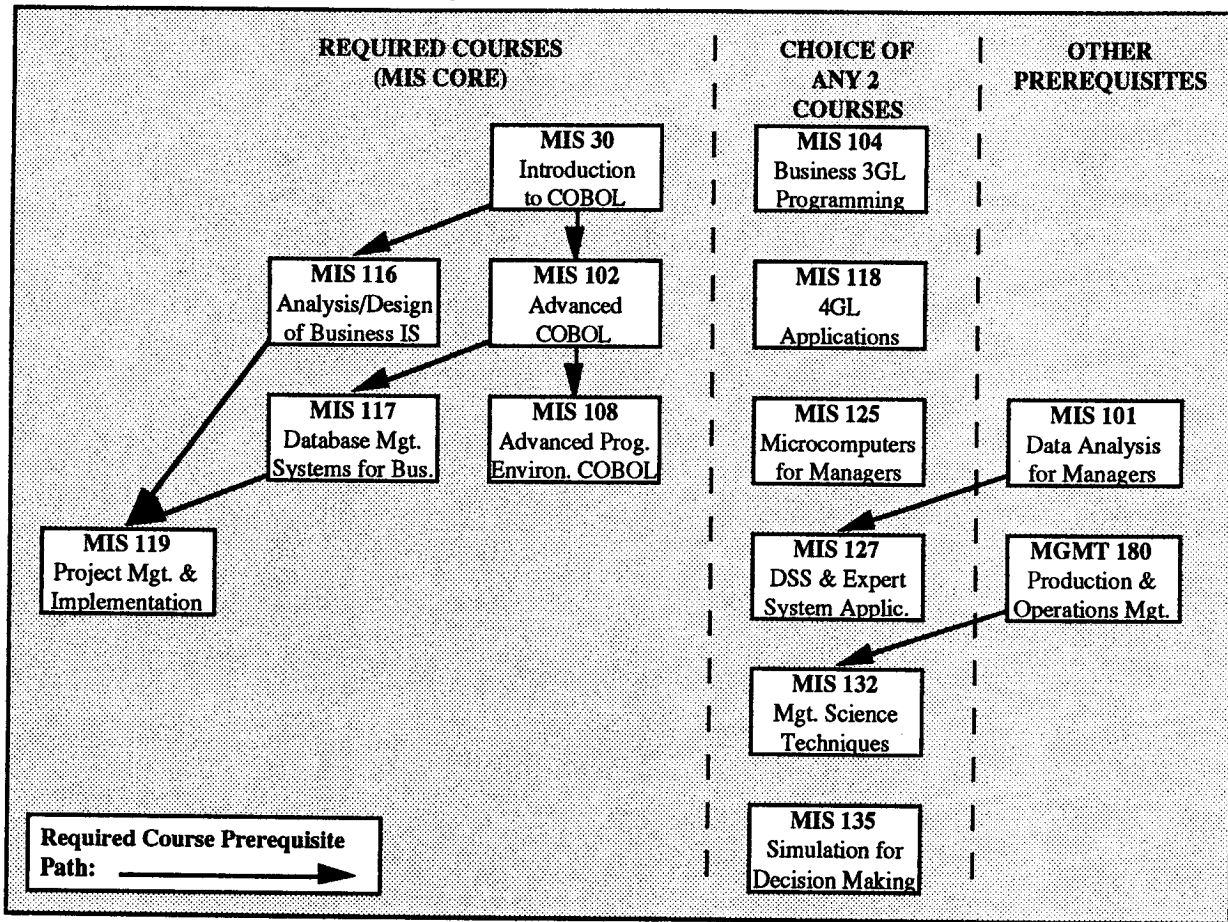
the 24 semester unit Management Information Systems major concentration within the B.S. of Business Administration. We currently have approximately 200+ undergraduate MIS majors. Our department also offers computer literacy, information system concepts, statistics, and management science courses in the School of Business Administration. Employers of our recent graduates include: EDS, Fireman's Fund, Foundation Health, Lawrence Livermore Labs, Pacific Gas and Electric, Price Waterhouse, State of California, and several local consulting firms.

Our major core consists of one year of COBOL, an advanced COBOL environments course (e.g., CICS), systems analysis and design, database, and a project implementation course. With the exception of the first year of COBOL, all of our advanced core courses include group project requirements. For the systems analysis and design, and project implementation courses, this requires working with actual systems for clients in the business community.

We are constantly evaluating our program in a continuing effort to improve it. For instance, the MIS 104 course has been, up until now, tied strictly to BASIC. We are going through the process to change it to a general 3GL microcomputer based course, where students could gain exposure to C and/or Pascal in either a traditional or object oriented environment.

We are just beginning to evaluate a proposal to improve the systems analysis and design and project implementation courses by offering a one year sequence which would replace the current

Figure 1. MIS Undergraduate Program @ CSUS †



† per 1992-1994 CSUS catalog.

structure. This sequence would allow more time to teach the basic analysis and design concepts before students interface with real clients. This sequence would also effectively add 2-3 weeks of usable time to the student's group projects by eliminating the need to seek out another client or to investigate a new problem in the second semester (where groups have changed and/or a different client/problem is more amenable to implementation).

### 3. IE Mapping.

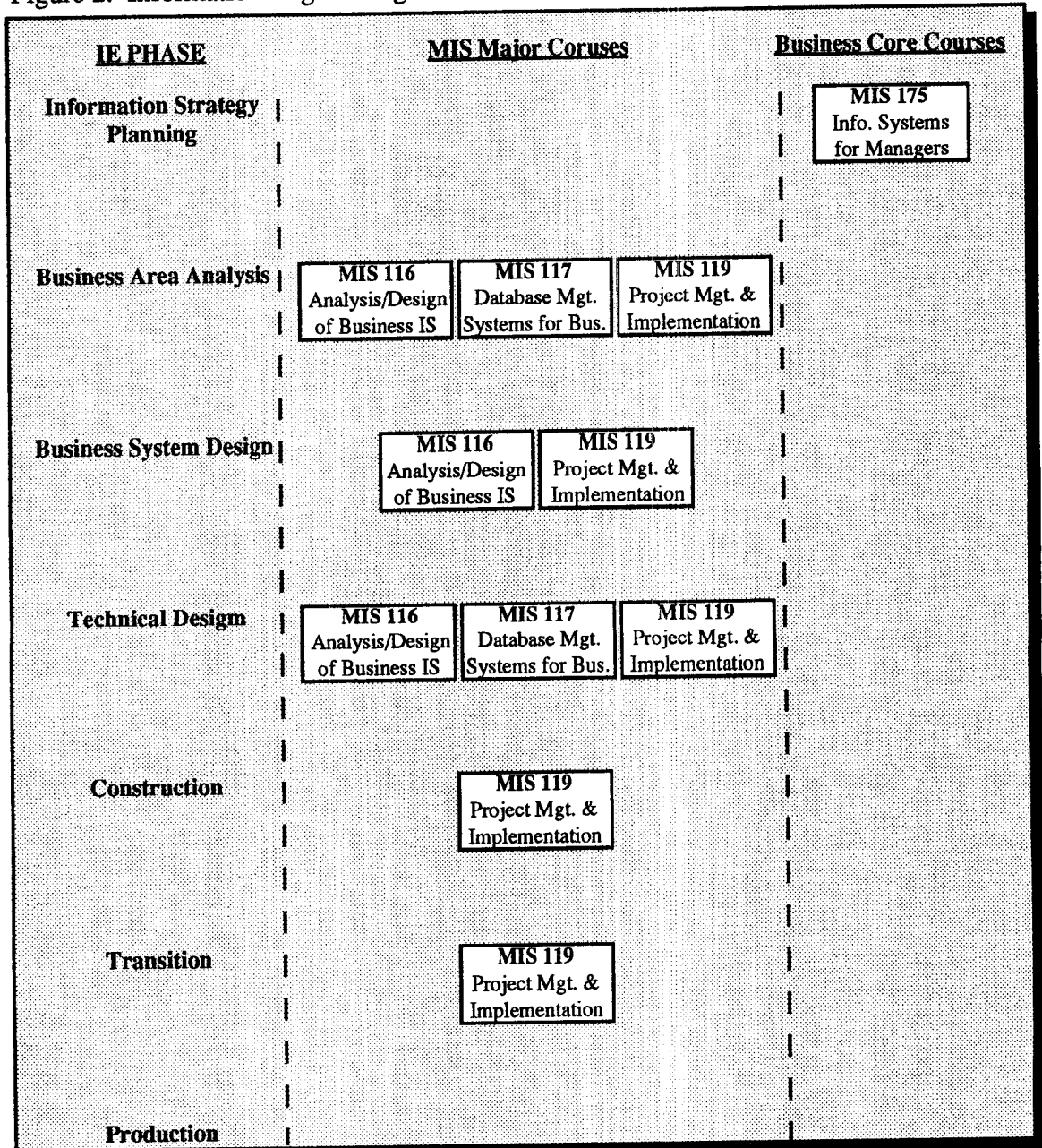
The introduction of the IEF in the new ICASE lab to our curriculum can best be described in terms of the methodology to which IEF is tied. This is shown in Figure 2. Naturally, the greatest impact would be in the analysis and design course, where we currently

employ the use of the Visible Analyst CASE product.

One problem, however, is that our current course follows the traditional Yourdon-DeMarco process oriented top-down systems development life cycle approach. This can be found in most common analysis and design textbooks (e.g., [1], [2], [5]). IE as a methodology can easily be introduced as a comparative methodology, but may never become the core methodology taught. Our students must be able to adapt to any environment in which they find themselves. How to create, read, interpret, and understand the documents associated with either methodology is a necessary skill set for our students.

The transition in our database course will be much easier. While the IE methodology is balanced in its process versus data orientation, this author believes that the data modeling is

Figure 2. Information Engineering Phases and MIS Courses



what drives the methodology. Currently, Entity-Relationship diagramming is currently taught as a data modeling approach. Converting these diagrams to relational schemas is also in the necessary skill set for our students. IE and IEF are easily integrated into this course.

Our project implementation course is our capstone course. Theoretically, our students apply their entire skill set to a problem. That skill set would include just about the entire IE methodology. Most groups develop their

systems using a 4GL such as Paradox, R:BASE, dBase IV, etc. With an ICASE lab available, it is expected that a large number of students would design and generate their systems using IEF.

The MIS 175 course, which is require of all business majors, is a senior level information systems course. A recent change split the old sophomore level computer literacy course into a freshman level productivity package literacy course (with DOS, spreadsheets, and word

processing), and the senior level information systems concepts course. This was done to address the issue of discussing information technology based solutions to business problems with students who had no context of the core business functions and/or issues. In this course all business majors are presented with Information Strategy Planning for several weeks.

#### 4. CASE Environments Course.

As with most large and powerful software tools, IEF has a very complex environment. This can be a very difficult problem to overcome. If we were to directly introduce IEF to our systems analysis and design course, too much of the limited lecture time would be taken up with questions about the environment and the effect of certain key strokes or button clicks.

The author is currently drafting a course proposal which would lead to an elective course (called CASE Environments) which would be taken prior to the systems analysis and design or database courses. The purpose of this course would be to acquaint the students with the IEF (and possibly other) ICASE software packages. This course would have a prerequisite of the first semester of COBOL, and could be taken concurrently with the second semester of COBOL.

The anticipated course methodology would be to focus on how to create, read, and generate the documents and logic necessary to develop a small simple system. The course would actually be very similar to the Rapid Application Development Tutorial that has been developed by Texas Instruments [4]. The data, processes, and logic are all provided to the student for a small system. The student can focus on learning the IEF environment. The issues related to the different development methodologies and diagramming techniques can be deferred until the student takes the systems analysis and design course.

This semester, one of the author's graduate students is developing an appropriate model and 'how to guide' which would be used in the proposed course. The author is also supervising an undergraduate MIS major in an independent study (which is designed to be a prototype for the proposed course).

#### 5. Related Issues.

Now that the ICASE lab is becoming a reality; opportunities, problems, and tough choices are greatly expanding for the MIS Department. The ICASE lab affect many areas. These includes our faculty, course content, relationships with industry, and future programs.

In the area of faculty, the author is currently the only 'IEF literate' professor in the department. One of our systems analysis and design professors, who was instrumental in this process and equally 'IEF literate,' has left our department. However, we are interviewing candidates for that position and we can now use the ICASE lab as a recruiting tool. We can also clearly describe our expectations of IEF related involvement to our candidates. While some of the other professors are reluctant to explore this new and exciting area, several others are interested in becoming 'IEF literate.'

A major issue is the perception of Information Engineering as an 'alternate' methodology. The mainstream methodology in MIS programs is still the process driven systems development life cycle. We do not believe that we can abandon this and focus our curriculum on the IE methodology, even though there will be a rising demand of IE literate analysts. There is an expectation that our graduates know the traditional methodology and the traditional techniques (e.g., data flow diagramming).

The process that we have been through over the last year has dramatically changed our view toward relationships with industry. These positive changes are fortunate in the continuing saga of restrained and declining budgets. In addition to closer ties with the ICASE lab contributors (Hewlett-Packard, Intel, and Texas Instruments) we are building closer relationships with Sutter Health, County of Sacramento, Deloitte and Touche, California Department of Motor Vehicles, and several other firms in our region. These firms are all investing to some degree in IEF, and will be looking to us in many ways to provide their employees with IE exposure or look to us as a source for new employees.

This leads to the one of the greatest opportunities for our department. Since firms are looking to us for IE related education. The demand for continuing education programs and certificate programs will dramatically rise. While

this will be secondary to our undergraduate and graduate degree programs, this could be very lucrative for our department.

## 6. Summary.

It will take several years to develop a curriculum which one could call IEF integrated. The first phase is to map IE and IEF to our existing courses. The second phase is to develop new courses to assist the transition. Eventually, we will reach a third phase in which we redesign our courses and curriculum, if necessary, to accommodate our environment. This environment includes: the Academy, the Faculty, the Community, the Students, the Technology, the Competition, and the Organizational Constraints of the institution [3]. Our department is simultaneously blessed and cursed with being the first in California to enter the IEF Academic Program. It is very exciting, but we will need to move rather slowly in order to insure success of this endeavor.

## 7. References.

- [1] DeMarco, T., Structured Analysis and System Specification, Yourdon Press/Prentice Hall, New Jersey, 1978.
- [2] Martin, Merle P., Analysis and Design of Business Information Systems, MacMillan Publishing, New York, 1991.
- [3] Sandman, T., "A Framework for Adapting a MS/MIS Curriculum to a Changing Environment," Forthcoming in Journal of Computer Information Systems.
- [4] Texas Instruments, IEF Development Tutorial: IEF Rapid Application Development/Tutorial Module, Texas Instruments Incorporated, 1991.
- [5] Yourdon, E., Modern Structured Analysis, Prentice-Hall Publishing, New Jersey, 1989.

# INFORMATION ENGINEERING, IEF, AND THE CIS CURRICULUM

Alden C. Lorents and Greg Neal  
Northern Arizona University, College of Business 15066  
Flagstaff Arizona 86011  
602.523.3510 602.523.7364

## ABSTRACT

The traditional life cycle approach to developing systems has been used since the 1950's, and continues to be used heavily today. There has been some progress in the development of tools to assist developers in the various phases of the life cycle. Most of these tools are not integrated nor do they have much ability to support the maintenance of existing systems. Much of the construction phase is still done with third generation languages, without much help from generators. It appears that integrated CASE (ICASE) tools are beginning to evolve that will support information engineering activities that start with business process engineering or re engineering, and progress through the analysis, design, construction, and implementation of systems. These tools will allow developers to spend more time on analysis and design activities and less time on construction activities. A big payoff of ICASE in the long run is the ability to maintain the developed systems at the system specification level, instead of the program level. The purpose of this paper is to explore ideas of how to integrate information engineering and an ICASE tool like IEF into the CIS curriculum. We will look at how IEF can support the learning objectives of various CIS courses. We will also look at how a CIS curriculum may change during the 1990's to produce graduates that can compete for jobs in industries using information engineering and CASE tools.

## INTRODUCTION

Many CIS curriculums over the past decade have been oriented toward producing entry level programmer/analysts with instruction in COBOL, database, systems analysis and design, and mainframe operating systems. Most of the course work in these programs tend to have two major trusts. The course work tends to concentrate on building NEW larger mainframe based COBOL systems, and most of the course work concentrates on the construction phase of the system life cycle. Technology changes are taking place that will require CIS programs to make changes to prepare their graduates for different entry level positions. These technology changes include ICASE tools that aid in the analysis and design, maintain the specifications, and generate the code for new systems. These systems can be an integration of graphical user interface (GUI) front ends, client-server architecture's, mainframe database systems, and process logic that is a combination of

generated code and legacy code. The legacy based systems include the 70 billion lines of COBOL code along with the IMS, IDMS and VSAM database systems that will continue to be around for a long time. The challenge is how do we integrate new information engineering (I.E.) technology into our CIS courses with the proper mix of the old and the new? When if ever do we stop teaching COBOL? How can a tool like IEF be used in various CIS courses to facilitate the teaching of information engineering concepts over the full range of I.E. activities?

## USING IEF IN THE SYSTEM ANALYSIS AND DESIGN COURSES

CIS curriculums across the country have been using some CASE software such as Excelerator, Brief CASE, and EASYCASE for a number of years. These tools have supported some of the upper CASE functions such as data flow diagramming, data modeling, and some model

verification between the data models and the process models. Most of these tools do not support detail process modeling, nor do they connect very well with tools that do. Consequently there is often a manual link between design and construction. Integrated CASE tools like IEF provide tools starting with the business planning model that have linkages throughout all phases of development including construction and implementation of the information system. Designs are done in enough detail so that code can be generated for the database system, the forms system, the COBOL or C programs, and the JCL to install and run the system. Examples of tools included in IEF that support these various phases of development are as follows:

**Figure 1**  
**Entity Elementary Process Matrix**

Elementary Process	Entity Type				
	EMPLOYEE	DIVISION	DEPARTMENT	PROJECT	TEAM
DELETE TEAM					
MODIFY TEAM					
ADD TEAM					
DELETE PROJECT	U			D	R
MODIFY PROJECT	U			U	

**Planning Toolset Diagrams**

- Function Hierarchy Diagram
- Function Dependency Diagram
- Business Function/Entity Type Matrix
- Subject Area Diagram

**Analysis Toolset Diagrams**

- Entity Relationship Diagram
- Process Hierarchy Diagram
- Process Dependency Diagram

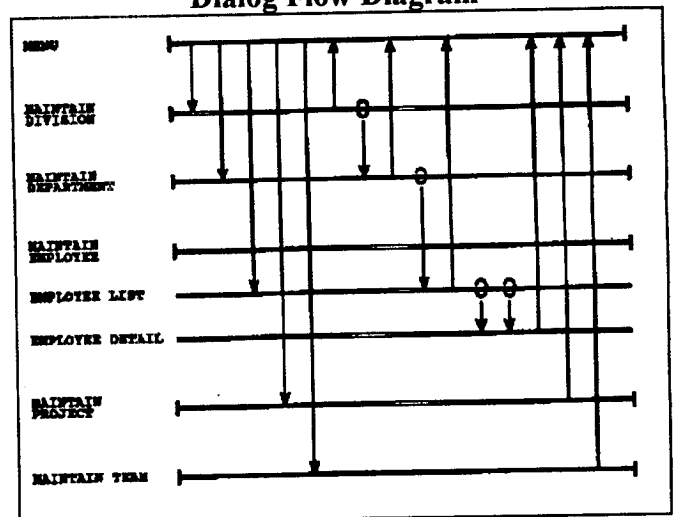
**Process Action Diagram**

**Business System Design**

- Screen Design Diagram
- Dialog Flow Diagram
- Window Design
- Procedure Action Diagram
- Structure Chart

An example of an Entity-Elementary Process Matrix is shown in figure 1 and an example of a Dialog Flow Diagram is shown in figure 2.

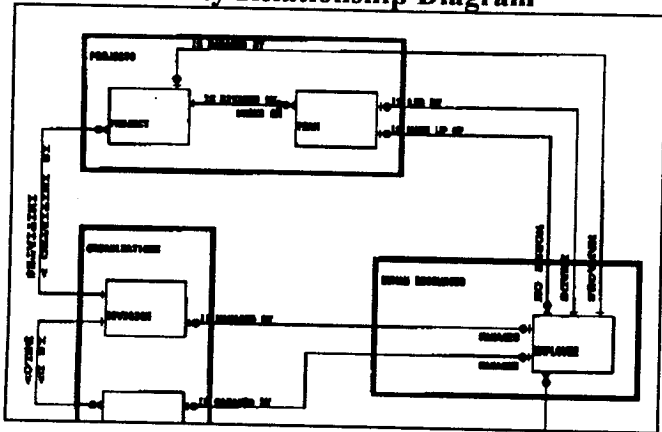
**Figure 2**  
**Dialog Flow Diagram**



**USING IEF IN DATABASE COURSES**

Data modeling is a central part of any system development. An essential component of any database course is to gain experience in setting up data models for various applications. IEF supports data modeling using entity relationship diagrams along with the related detail definitions of each entity and each relationship. Students can use the tools to set up referential integrity definitions, primary keys, domain constraints, attribute definitions and entity relationships. A sample entity relationship (ER) diagram is shown in figure 3.

**Figure 3**  
**Entity Relationship Diagram**



IEF supports the data modeling activities at various phases of the information engineering process. During the planning stage IEF supports data oriented matrices, subject area/entity relationship diagrams and a subject area list shown in figure 4. During analysis and logical design the entity relationship diagrams and lists are used. Once the logical data model is completed the ER diagram can be converted to a data structure diagram to detail how the database will be converted to a physical database. This may include some de normalization and other modifications to address performance issues.

**Figure 4**  
**Subject Area List**

Type	Name
Subject Area	IEF 5.1 INSTALLATION MODEL
Subject Area	HUMAN RESOURCES
Subject Area	ORGANIZATIONS
Subject Area	PROJECTS

The construction stage builds the data description language (DDL), which includes all the create statements of the target relational database. Some sample DDL that has been generated is shown below in figure 5. The generated DDL includes the creation of the tables along with the related referential integrity, domain constraints, and index tables. IEF allows the developer an option of building referential integrity into the database

system or into the procedure code, depending on the ability of the database system to support referential integrity.

**Figure 5**  
**Sample Generated DDL**

```

CREATE DATABASE "IEFSUP40" ;

CREATE TABLE "IEF50_TEAM"
("NAME" CHAR(25) NOT NULL,
"NUMBER" SMALLINT NOT NULL,
"FK IEF50_PROJECTNUM" SMALLINT NOT NULL,
"FK IEF50_EMPLOYNUM" INTEGER
PRIMARY KEY
("FK IEF50_PROJECTNUM"
"NUMBER"
));

COMMIT ;

CREATE INDEX "ID000061" ON "IEF50_TEAM"
("FK IEF50_EMPLOYNUM" ASC);

COMMIT ;

CREATE UNIQUE INDEX "TEAMID" ON "IEF50_TEAM"
("FK IEF50_PROJECTNUM" ASC,
"NUMBER" ASC);

COMMIT ;

```

The IEF tool has the ability to do consistency checks at the various stages to make sure that all relationships, entities, and fields have been properly defined with the rest of the system. Some of the consistency rules used are shown in figure 6.

**Figure 6**  
**Sample Consistency Rules**

- Entity Types
  - Each entity described by an entity type must be uniquely identifiable.
  - An entity type must have at least one attribute or two relationship memberships.
  - An entity type must participate in at least one relationship.
  - An entity type is part of one and only subject area.
- Relationships
  - Each relationship associates one or two entity type and depicts a pairing between exactly two entities.
  - A relationship must not have attributes.
  - An optional relationship membership may not participate in an identifier.



Other uses of IEF in the database course could include the following:

Reviewing generated SQL code that is embedded in the COBOL procedures.

Reviewing the data dictionary that supports the database manager under OS/2 to view the definitions setup by the DDL.

Making a change in some data specifications or in an E-R diagram, and following how those changes affect DDL, the data dictionary, and embedded SQL code.

Comparing the differences between logical and physical data models.

### USING IEF IN PROGRAMMING COURSES

A CIS program that emphasizes information engineering with the intent of putting out information analysts, information engineers, and information system developers using integrated CASE tools will probably begin to cut down on the programming content of the curriculum. However, some programming content will always be required because the analyst/designer will always have to use some specification language to define procedural solutions. Students need to have some practice in solving problems using a procedural language. Another concern is the need to be somewhat familiar with COBOL because of the amount of COBOL that is in legacy systems that will be around for a long time.

A first course in COBOL will probably be a part of this CIS program for the remainder of the 1990's. However, we could see changes taking place in the advanced COBOL course that would include program design issues using the process hierarchy and process dependency diagrams shown in figures 7 and 8. Students would develop process action diagrams for a number of elementary processes such as create order, change order, delete order,

and process order. An example of a process action diagram is shown in figure 9. You can see that developing process action diagrams requires a fair amount of procedural logic problem solving ability. However, object oriented process development may change this approach some. Most IEF action diagrams are developed partially from the graphical diagrams and support matrices like the one shown in figure 1. By using a feature called process synthesis the process logic is developed through an intelligent interactive dialog with the student based on previous decisions made in data modeling. Program action diagrams that are more complex and not as common would have to be filled in by the student. This is done through menus and within the context of the process and data entities involved. Only syntactically and semantically correct statements are allowed.

Figure 7  
Process Hierarchy Diagram

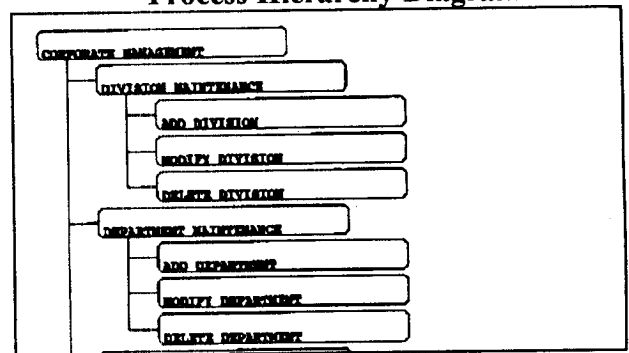
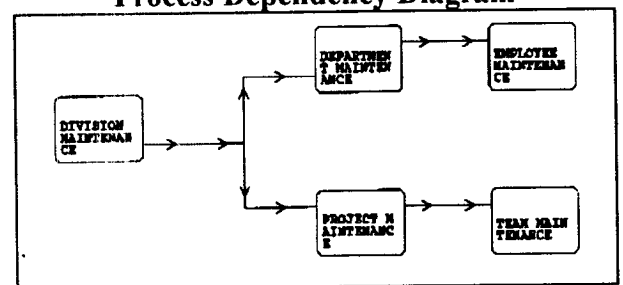
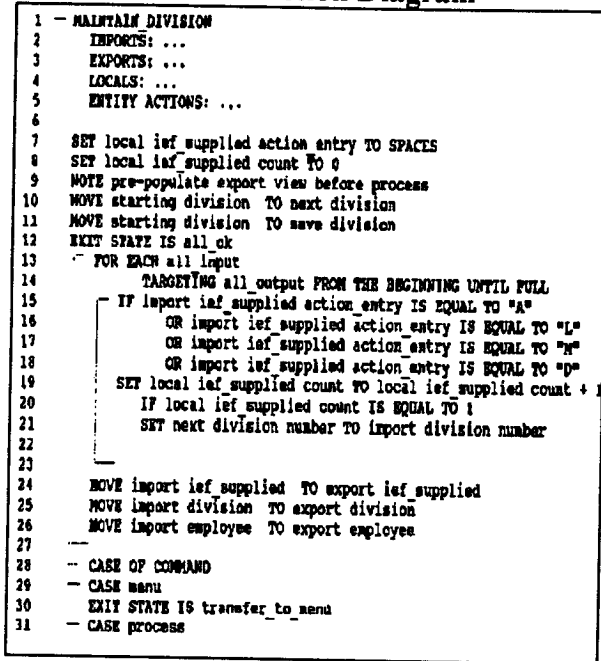


Figure 8  
Process Dependency Diagram



**Figure 9**  
**Process Action Diagram**



**Figure 10**  
**Sample Generated Code**

```

/* strtok the new compressed string into temp_tc, temp_cmd,
and temp_rest
if (token = strtok(unformatted_string, " "))
{
  memcpy(temp_tc, token, strlen(token));
  if (token = strtok(0, " "))
  {
    /* dialog flow, unformatted input, reset, or restart */
    memcpy(temp_cmd, token, strlen(token));
    application_list.unformatted_data.sw = UNFORMATTED_DATA;
    memcpy(temp_rest, token + (strlen(token) + 1),
           strlen(token) + (strlen(token) + 1));
  }
}
else
{
  /* input was transcode only */
  memcpy(temp_cmd, default_cmd, sizeof(temp_cmd));
  default_cmd_used.sw = pos;
}
}

if (t_known_dialog_action())
{
  application_list.unformatted_data.sw = neg;
  if (memcmp(temp_cmd, T_HELP_RESTART_D, CMD_LEN) == 0 ||
      memcmp(temp_cmd, T_HELP_RESTART_E, CMD_LEN) == 0)
  {
    /* help restart */
    if (memcmp(temp_rest, spaces, sizeof(temp_rest)) == 0)
    {
      /* help restart no data */
      memcpy(temp_cmd, restarttp, sizeof(restarttp));
    }
  }
}
else
  
```

Other topics that could be appropriate for the advanced programming course includes developing screens and a dialog flow diagram. A major portion of developing a system is the dialog between the screens and the procedures. An example of a dialog flow diagram was shown in figure 2. Procedure synthesis can be used to automate the creation of logic for menus and selection lists. Students can be given assignments to add logic to an existing model and see the examples of the generated COBOL code along with the embedded SQL. An example of this generated code is shown in figure 10.

### USING IEF IN THE MIS COURSE

A number of business schools teach an MIS course to all business majors as a part of the common body of knowledge. It is important in this course to give students an understanding of the relationships between business systems and the information systems that support these business systems. It is also important for students to gain an understanding of the various components of an information system and how these components

relate to each other and go together to build an information system. This could include activities on determining what the business is, what information requirements are needed to support the business, and how the information is related to the critical success factors of the business. Students can gain this understanding by working on projects using a number of the IEF tools. These tools could include various planning matrices, an entity relationship diagram, a process dependency diagram, window design and prototyping, process hierarchy diagram and some introduction to process action diagrams. Students could fill in parts of completed models, and perhaps go through code generation and see the results of the system in operation. Students could work in teams using information systems majors as consultants.

IEF also supports some tools for business process re engineering, and is working on further development in this area. The MIS course would be an appropriate course to introduce the concept of business re engineering and analyze the impacts on information and system design.

## **A PROPOSED CIS CURRICULUM MODEL USING IEF**

The following CIS curriculum is proposed to produce graduates with a heavy emphasis in information engineering. The graduate would be targeted toward larger organizations who currently use a lot of mainframe, COBOL, IMS/IDMS/DB2, who plan to use CASE and client-server environments. The student would get a Business degree that includes the typical introduction to CIS as well as an advance IS course that includes an introduction to business planning, business re engineering, business area analysis using some of the planning and analysis toolsets from IEF.

The Information Engineering Emphasis would include the following 21 to 24 semester hours where each course is three semester hours.

### **CIS 220**

**Information System Design I**  
Introduction to program design  
Introduction to COBOL  
Micro Focus Workbench  
ISPF/SPF-PC

### **CIS 310**

**Information System Development I**  
Introduction to data modeling  
Introduction to relational DB  
SQL  
Database maintenance  
(create, read, update, delete)  
Data dictionary maintenance  
Database creation  
COBOL/SQL  
IEF Data modeling  
IEF Program action diagrams  
IEF dialog flow  
IEF screens  
IEF construction

### **CIS 320**

**Information System Design II**  
Advanced COBOL  
Advanced table handling  
IEF dialog flow  
IEF screen design  
IEF program action diagrams  
IEF design toolset

**IEF construction**  
**Re engineering legacy systems**

### **CIS 330**

**Information System Development II**  
Business area analysis  
Business system design  
Business re engineering  
Technical design  
Prototyping  
Planning toolset  
Analysis toolset

### **CIS 440**

**Information System Development III**  
System project course  
Re engineering legacy systems  
IDMS/IMS database interfaces  
Rebuilding an existing system  
Client-server with hooks to IMS/IDMS  
IEF technical design  
IEF construction

### **CIS 460**

**System Development Topics**  
Client-server development  
Oracle/Sybase/SQL server  
4GL  
Report generation  
GUI interfaces to SQL/servers

### **CIS 450**

**Operating Systems (Possible elective)**  
OS/2  
Windows  
UNIX  
MVS  
Novell networks

### **CIS 470**

**Telecommunications and Networking**  
Introduction to telecommunications  
EDI  
Network design  
Setting up and using Novell networks  
PC/TCP  
E-Mail systems  
Multi-media systems

## SUMMARY

CIS curriculums that have targeted their output at entry level programmer analyst positions with COBOL, database, and mainframe orientations will have to make changes during the 1990's to stay competitive. One of the ways to make the change is to move in the direction of an information systems engineering emphasis using integrated CASE tools. As this change is made programming courses will concentrate more on design and less on language. There will be more emphasis on tools that will aid the analysis and design process. More time can be spent on the relationships between the information model and the business model. There will be more integration of database courses with analysis and design courses. There will continue to be controversy over how much programming language should be taught. COBOL, IMS and IDMS will continue to be around for a long time. COBOL versus C will continue to be an issue. Program design and problem solving using both procedural and non-procedural approaches will continue to be important. Using some language support such as COBOL, C, and SQL to support program design and problem solving serves multiple objectives.

## REFERENCES

Business Design Facility - Effective Software for Business Process Re-Engineering, Texas Instruments, August, 1992.

IEF for Client/Server Applications, Texas Instruments, September, 1992

Information Systems - The DMPA Model Curriculum for a Four Year Undergraduate Degree, Data Processing Management Assn., 1991.

IEF Technical Description..Methodology and Technology Overview, Texas Instruments, August, 1992.

IEF Basics Handbook, Texas Instruments, December, 1991.

A Guide to Information Engineering using IEF, Computer-Aided Planning, Analysis and Design, Texas Instruments, 2nd Edition, January 1990.

IEF Statement of Direction, Texas Instruments, 1991.

## Factors to be Considered in Adopting I-CASE Undergraduate Computing Curricula

Claude L. Simpson, Jr.  
Debasish Banerjee

Northwestern State University of Louisiana

### Abstract

Northwestern State University was the first undergraduate program to use the Texas Instruments' Information Engineering Facility (IEF™). This paper summarizes some of the factors that were important in the development of an I-CASE curriculum.

### Why consider I-CASE?

Shifting demands in industry for CIS majors led us to believe that employment opportunities for our CIS graduates were not good and did not look promising for the foreseeable future. In fact, in the last three years, we have placed three graduates who are directly working in the CIS area.

One of the authors of this paper recently received a call from a potential employer, after we announced the Information Engineering program, who stated that he would hire an entire graduating class if they were trained in Texas Instruments IEF™ tool. We were unable to fulfill this request. We will, however, be able to satisfy such requests in the near future.

### What tools are available?

There are a number of I-CASE tools available today. A number of these include EXCELERATOR™, POSE™, IEW™, IEF™, and many others. The faculty at NSU had used a number of these for many years and not been completely satisfied with them for a variety of reasons. The primary reason was the price of the products vis-a-vis the capabilities of the product. For example, AEW™ and IEF™ were capable of doing what we wanted but we felt that we were not able to afford the costs of these products. Some of the other products that we could afford did not offer all the facilities that we desired.

### What is involved--benefits and responsibilities

Although the efforts made in our behalf by our alumni were probably from a sense of loyalty to NSU, there are substantial benefits that accrue to all sides of the partnership. From the university view, the benefits include funds for participation in the IEF educational program, equipment donations by the patrons, faculty training, and a positive image among our sponsors. From the patron's view, the university provides co-operative education assistance to the patron in selecting students for participation in an internship program at the sponsors location. The patron is afforded the opportunity to view potential employees in a work environment without having to make a major financial expenditure for training an Information Engineer. From the student's view the student is allowed the opportunity to earn funds to further his/her education and to see how well he/she fits into the work force of the potential employer.

### Training of Faculty

Training of faculty in the use of a specific Information Engineering tool is a critical factor

in the success of an I-CASE program.

The Texas Instruments IEF™ co-operative training program

As a part of the University Program, Texas Instruments has committed to make training available to university program participants on a no-charge, space-available basis. The NSU faculty have attended a considerable number of classes under the program. The courses, the faculty, and the facilities are excellent.

Based on the current fee schedule, this amounts to a major contribution to the NSU Information Engineering efforts.

Modification of Curriculum

The major problem in adopting an Information Engineering curriculum is the modification of an existing curriculum or the modification of an existing curriculum. NSU chose to modify the existing CIS curriculum to include Information Engineering courses within existing courses and to add a minimal number of new courses. The primary reason for this course of action was university politics.

The typical CIS curriculum

NSU's CIS curriculum was the typical Data Processing Management Association (DPMA) Model Curriculum. This curriculum includes the traditional programming courses, systems analysis and development courses, telecommunications, and others.

The I-Case curriculum

The I-CASE curriculum at NSU is:

<b>Course Name</b>	<b>Course Description</b>
Programming Concepts	This course covers the basic programming development cycle and includes IEF™ concepts.
Introduction to Information Engineering	This course gives an overview of the Full Life Cycle of the IEF™. James Martin's Info Engineering I text is used in this course.
Introduction to COBOL	Information Engineering tools and concepts are used to develop solution algorithms for lab assignments.
Information Engineering I	Students are strongly trained in the Information Strategy Planning and Business Area Analysis portions of the IEF™. James Martin's Info Engineering II book is used here.
Advanced COBOL	Continued use of IE methodologies to develop program solutions in this course.

Information Engineering II	Covers the Business Systems Design and Technical Design Construction portions of the IEF™.
Database Methods	Focuses on the IEF™ use of databases and uses DBM that is included in OS/2 Extended Services.
Operating Systems	Focuses on the OS/2 operating system and host system interfaces with the IE tool.
Applied IEF™ Project	Capstone course of the curriculum. Student applies knowledge of the Information Strategy Plan, Business Area Analysis, Business Systems Design and Technical Design/Construction. This course is planned as a cooperative education endeavor.
Telecommunications	Very little change made in this course. Very little Information Engineering covered here. Left in curriculum because it was judged to be an important technology course for students.
Artificial Intelligence/ Expert Systems Course	This course covers AI/ES using tools such as Neuralware™. The course was added to the curriculum because it was judged to be an important technology course for students.
Decision Support Systems	This course to be added to curriculum and integrated with the IEF™ to promote understanding of DSS.

#### Summary

The Information Engineering program is underway at NSU and is currently enjoying considerable popularity among students, patrons and the faculty. The CIS curriculum has been revised to include the IEF™ tool as the tool of choice in implementing an Information Engineering curriculum at Northwestern State University. Equipment has been installed, rooms renovated, and software installed and tested. The faculty are well on their way to completing an ambitious training schedule in the use of the IEF™ tool and in Information Engineering concepts. Support has been received from corporate partners and considerable help has been received from them on the revision of the Information Engineering curriculum.

**DO AS I SAY DO: SOFTWARE PIRACY  
AS A FAILURE OF ETHICAL WILL**

Diane M. Miller  
University of Southern Mississippi, Gulf Coast

**Abstract**

There is an apparent discrepancy in the academic community between attitudes toward software piracy and actions taken regarding it. This study finds correlations to academic major, implying that information systems majors may be simultaneously more aware of and less committed to a code of ethics regarding software piracy.

**Introduction**

Students face a new category of legal and ethical issues as they incorporate computer proficiency into their business training. One such issue is that of the defensibility of making unauthorized copies of software.

There is no doubt that piracy is illegal. Amendments covering software were added to the Copyright Act of 1976 back in the 1980s [3]. Late in 1992, a specific Software Piracy Act was signed into law, elevating commercial software piracy to a felony and imposing prison terms of up to five years and fines of up to \$250 thousand for anyone convicted of stealing at least 10 copies of a program, or more than \$2,500 worth of software [8].

There is also no doubt that such copying is widespread: Bootlegging results in an estimated annual loss of \$1.2 billion in software sales, compared to annual sales of \$6 billion to \$7 billion. These figures support the estimate that one in five personal computer programs in use today is an illegal copy [8]; other sources place the estimate as high as ten unauthorized copies for each legitimate copy [6]. The applicable dollar figure for

maximum punishment under law is well within the range of the value of software found on many personal computers.

There is, however, an obvious discrepancy between law and practice. As the training ground for professionals, business schools have a particular interest in and responsibility toward the problem of software piracy. This study examines student attitudes toward software piracy as compared to student involvement in the act of piracy.

**Perspectives on the Ethics of Copying Software**

Several studies have focussed upon the coexistence of widespread copying with known illegality, especially in an academic environment [1, 2, 4, 5, 6, 7]. One explanation for the phenomenon is that copying is not perceived as unethical, even though it is known to be illegal. As Gray and Perle observe, "If an activity is not perceived to be unethical, it is very difficult to . . . eliminate it through aggressive enforcement" [4, p. 27]. According to their findings, for example, 64.9% of business faculty members think copying software is illegal but



only 55.6% think it is unethical. They also found that faculty members are more likely to consider copying justifiable when the software is for personal use rather than for the use of someone else.

Other research indicates that copying persists even when the perpetrators consider the behavior unethical. Shim and Taylor surveyed 500 randomly selected business faculty. They report that 66% of their 218 respondents consider copying unethical. However, 70% of the respondents admit making unauthorized copies, and 90% believe that their colleagues copy [7].

Not surprisingly, copying extends to the student population also. Cohen and Cornwall found that 86% of students surveyed agree that most students copy software instead of purchasing it [1]. Christoph, Forcht, and Bilbray report similar results [2]. Reid, Thompson, and Logsdon found that the majority of student respondents held attitudes that are inconsistent with the law [6]. Miller suggests that the academic environment may be partially at fault, that the educational process itself is a source of some attitudes accounting for the discrepancy between knowledge and actions [5].

Shim and Taylor's findings [7], cited earlier, imply that a sizable portion of the faculty members surveyed must be deliberately suppressing their own best understanding of the ethics of unauthorized copying. If similar attitudes and actions are found among students, the question remains as to whether the level of exposure to the use of computers in academia increases the likelihood of a knowledge/action discrepancy in dealing with software piracy. The present study will compare the

attitudes and reported actions of information systems majors with those of non-majors as a measure of possible differences in attitudes and actions attributable to increased exposure to the use of computers in an academic environment. The formal hypotheses may be stated as follows:

H1: There will be no difference in attitudes toward software piracy between survey groups based on major.

H2: There will be no difference in actions involving software piracy between survey groups based on major.

## EXPERIMENTAL METHOD

### Task

Subjects completed a questionnaire reporting their attitudes toward the situations described in several scenarios in which software piracy had taken place. For each case, subjects were asked to select as many listed circum-stances as they considered acceptable reasons for making unauthorized copies, and to supply additional reasons if needed. The list included the following choices:

- To teach students
- To further research
- To provide a convenient working environment
- To save money
- Never
- Don't know

Assured of the confidentiality of their answers, subjects were further asked to report their own actions and the observed actions of others in similar situations. They were asked to categorically identify all those who they were sure had made unauthorized copies, using the list:

- Self
- Another student
- A professor
- Other university personnel
- No one

**Subjects**

The survey was conducted among upper-division students in business information system classes. Overall mean age was 29.44; age range was from 20 to 47. Student responses were divided into two survey groups according to whether or not the student was majoring in information systems. The subjects were drawn from among juniors, seniors, and graduate students at a regional university within the Southeastern United States. All students attended night classes only, indicating likelihood of employment. Majors were surveyed after they had taken at least two courses beyond the core, but before they received specific training in information system security. Other students had been educated in general information systems concepts only.

**Demographic Questionnaire**

To test for homogeneity of the survey groups, students answered demographic questions. The variables on the questionnaire were tested for use as covariates in the analysis.

**Independent Variable**

The independent variable manipulated in this study was student major. All students responded to the same questionnaire.

**Dependent Variables**

Two dependent variables were evaluated as measurements of possible differences in the major versus non-major survey groups.

The first measured attitudes toward software piracy, focusing upon respondents' agreement with the stated position that piracy is never acceptable versus identifying occasions when it may be acceptable. The second dependent variable tabulated personal admissions of having committed software piracy.

**RESULTS**

**Subjects**

Subject demographics are presented in Table 1. The demographic factors appear to be similar for the groups. Nevertheless, gender and employment status were used as covariates in the following analysis.

**Table 1  
Subject Demographics by Test Group**

	IS Majors	Non-Majors
<b>n</b>	59	74
<b>n by Gender</b>		
Female	25	33
Male	34	41
<b>n by Employment</b>		
Fulltime		
related job	31	37
Other		
employment	11	21
Not employed		
for pay	17	16
<b>Mean Age</b>	30.45	28.64

**Hypothesis 1**

The first hypothesis postulated that increased exposure to the use of computers in an academic environment, as measured by status as an IS major or non-major, would

make no difference in attitude toward the ethics involved in software piracy. Table 2 presents crosstabulated results. No majors responded that copying is justifiable to save money, but a greater percentage of majors than non-majors found copying acceptable if the intent was to instruct or learn. Well over half of majors responded that it is never justifiable to make unauthorized copies; less than half of non-majors responded so.

**Table 2**  
**Justifiable Occasions for Copying Software Identified by IS Majors Versus Non-majors (%)**

Justifiable Occasion	IS Majors	Non-majors
Teach students	38	34
Further research	15	14
Convenience	8	11
Save Money	0	14
Never	54	46
Don't Know	8	9

The results of the ANCOVA test of hypothesis 1 are presented in Table 3. Observations were treated in relationship to agreement with the response "Unauthorized copying is never justifiable" as the dependent variable. Responses of "don't know" were not considered in the analysis. This test indicates that major had a marginally significant effect on the incidence of reported belief that copying is never justifiable, and that the other

covariates (gender and employment status) did not have a significant effect. Since differences were found, H1 is rejected.

**Table 3**  
**ANCOVA of Responses on Attitudes**

	Sum-of-Squares	DF	Mean-Square	F-Ratio	P
Employment	19.03	2	9.52	2.03	.15
Major	16.55	1	16.55	3.34	.07
Gender	.99	1	.99	.2	.66
Error	643.89	127	5.07		

**Hypothesis 2**

The second hypothesis postulated that increased exposure to the use of computers in an academic environment, as measured by status as an IS major or non-major, would make no difference in whether or not students admitted actually engaging in software piracy. Table 4 presents crosstabulated results. All students reported similar experiences with unauthorized copying by faculty and other university employees. However, the percentage of respondents admitting making copies themselves was more than twice as great among majors as among non-majors. Similarly, the percentage observing another student copying was more than twice as great for majors as for non-majors. Only 15% of majors reported never having observed unauthorized copying, whereas almost half of non-majors claimed never to have seen it.

**Table 4**  
**Observed Actual Unauthorized**  
**Copying Reported by**  
**IS Majors Versus Non-majors (%)**

Observed Copying	IS Majors	Non-majors
Self	38	17
Another student	77	37
A professor	23	20
Other university personnel	6	6
No one	15	49

The results of the ANCOVA test of hypothesis 2 are presented in Table 5. Observations were treated in relationship to agreement with the response "I have personally made unauthorized copies of software" as the dependent variable. This test indicates that major had a significant effect on the incidence of reported personal unauthorized copying of software. In this case, employment status is also significant, although gender is not. Since differences were found, H2 is rejected.

**Additional Finding**

32 majors and 36 non-majors responded that copying is never justified. Isolating those observations, it was found that, among those responding that unauthorized copies are never justifiable, the percentage reporting that they have personally copied is 16% among majors and 3% among non-majors.

**Table 5**  
**ANCOVA of Responses on Attitudes**

	Sum-of-Squares	DF	Mean-Square	F-Ratio	P
<b>Employment</b>	128.48	2	64.24	6.42	.00
<b>Major</b>	769.25	1	769.25	7.72	.00
<b>Gender</b>	164.19	1	164.19	1.65	.2
<b>Error</b>	12650.5	127	99.61		

**CONCLUSION**

This survey grouped student responses by whether or not they were contributed by an advanced information systems major. The grouping was used as a measure to assess how increased exposure to computing in an academic environment impacts student attitudes and student behavior regarding the unauthorized copying of software. Both Hypothesis 1, that there would be no difference in attitude between groups based on major, and Hypothesis 2, that there would be no difference in actions, were rejected.

The results of the study found measurable differences between IS majors and non-majors in both attitudes and behavior, in reference to making unauthorized copies of software. Majors in the survey group were more aware of the general ethics and/or legalities which hold that unauthorized copying is never justifiable.

Simultaneously, majors in the group reported less commitment to a code of personal behavior which would

operate in accord with their awareness of ethics and legalities. Although academic major was used as a measure of exposure to the academic computing environment, it is recognized that the correlation is imperfect. Students with other business majors, depending upon background, might conceivably have encountered as much exposure. Most obviously, other factors besides exposure to the academic computing environment might also contribute to attitudes and to the willingness to act. Also, IS majors might simply be more willing to report their own unauthorized copying.

However, this research assumes that major is a reasonable measure of exposure to computing. Ostensibly, greater exposure logically leads to both greater ability to pirate software and greater opportunity to observe the act of piracy. This research finds some evidence that this exposure yields a greater understanding of the implications of piracy. Ideally, in the light of this understanding, greater exposure should also deter the practice of piracy, but these findings indicate that the opposite is true. In actual practice, majors are more willing, not less willing, to copy software.

Perhaps the sad truth is that we as mentors are failing our charges. Current curricula include a strong emphasis upon ethics, and indeed we stand before each class and say the appropriate words. Yet others cited [e.g. 4, 7] have found that professors do not behave in accordance with the ethical guidelines they teach. Students learn their lessons well. They may be finding in the example of their professors a mode of behavior that allows one to practice software piracy while cautioning others to avoid it.

## REFERENCES

- [1] Cohen, E., and Cornwall, L. "College Students Believe Piracy is Acceptable," CIS Educator Forum, (1:3), March 1989, pp. 2-5.
- [2] Christoph, R., Forcht, K., and Bilbray, C. "The Development of Information Systems Ethics: An Analysis," The Journal of Computer Information Systems, Winter 1987/1988, pp. 20-23.
- [3] "Copyright Protects Program's Structure, Sequence, and Organization," Computer Law and Tax Report, October 1986, pp. 5-7.
- [4] Gray, A. P., and Perle, R. J. "Attitudes on Legality Versus Ethics in Software Copyrights," Interface, (14:4), pp. 27-34.
- [5] Miller, J. E. "Computer Abuse: An Academic Perspective," Proceedings of the 12th National Security Conference, Baltimore, October 1989, pp. 615-618.
- [6] Reid, R. A., Thompson, J. K., and Logsdon, J. M. "Knowledge and Attitudes of Management Students Toward Software Piracy," The Journal of Computer Information Systems, Fall 1992, pp. 46-51.
- [7] Shim, J. P. and Taylor, G. S. "Business Faculty Members' Perceptions of Unauthorized Software Copying," OR/MS Today, March 1988, pp. 30-31.
- [8] Weise, E. "You Can Run, But You Can't Hide From 'Software Police'," Biloxi Sun-Herald, Wednesday, November 9, 1992.

**INFORMATION COMPILATION AND DISBURSEMENT...  
MORAL, LEGAL, AND ETHICAL CONSIDERATIONS**

Dr. Karen A. Forcht, Professor  
Dr. Joan K. Pierson, Professor  
Department of Information and  
Decision Sciences  
and  
Dr. Daphyne S. Thomas, Associate Professor  
Department of Finance and  
Business Law

College of Business  
Zane Showker Hall  
James Madison University  
Harrisonburg, VA 22807  
703-568-3057, -3014, -3070 (O)  
FAX: 703-568-3299  
BITNET FAC\_FORCHT@JMUUVAX1

**ABSTRACT**

**Information Compilation and Disbursement...  
Moral, Legal, and Ethical Considerations**

The widespread collection of data about individuals is increasing at dramatic rates as more and more organizations increase their desire for information in order to develop new markets for products, gain a competitive edge, or to expand their operations. The paramount concern today is not focused entirely on halting data collection but is primarily concerned with the disbursement and use of information.

Information and activity are being monitored and sold to marketers, credit bureaus, insurance companies, and government agencies, for their specific use. The misuse of this information and the concern for invasion of an individual's privacy are issues that are not being fully addressed and controlled.

Legislation to monitor this information transfer is lagging far behind the rapid growth and disbursement of technology. Measures must be instituted to regulate this activity so that information is used for its intended, acceptable purpose and not freely distributed.

**Panel - Implementation of the DPMA Model Curriculum**

**Chair: Prof. Kathryn McCubbin, Christopher Newport University**

**Panelists:**

**Dr. Eli Cohen, Eastern New Mexico State University**

**Dr. Larry Eggan, Southern Illinois State University**

**Dr. Mayes Mathews, Christopher Newport University**

**Dr. Longy Anyanwu, Georgia Southwestern University**

The objective of this panel will be to share our experience in the implementation of the DPMA 90 Model Curriculum. Dr. Mathews was instrumental in his institution selecting the model curriculum as its BSBA concentration in MIS and has taught four of the curriculum's suggested courses. Dr. Anyanwu has also taught model curriculum courses and will describe the various problems he has experienced. Drs. Eggan and Cohen are departmental chairmen and will describe the programmatic impact of the model curriculum. Panel discussion will focus on identifying the strengths and weaknesses of the model curriculum in practice including program assessment, effectiveness of the "spiral" pedagogical approach and textbook selection.

**Where is our Future in Business Schools:  
Information Technology or Business Processes?**

Steven Alter  
University of San Francisco  
McLaren College of Business  
San Francisco, CA 94117  
415-666-6383

**ABSTRACT**

In his editorial in the Dec.1992 issue of *MIS Quarterly*, Blake Ives quoted a senior IS executive describing the possibility that in the near future, "the IT function will be very much like an Alka-Seltzer tablet dropped in a glass. It will have gone away and will exist throughout the entire organization." Since most of our academic colleagues and many students use IT routinely, we face our own version of the Alka-Seltzer problem: If IT is everywhere and is easier and easier to use, what is our contribution?

In many business schools our main contributions to the generalist curriculum are packaged as one or several courses, each with major problems. The *Introduction to Word Processing and Spreadsheets* introduces skills that can be mastered in high school. The *Computer Literacy* course should wane as students who already use CD players, VCRs, answering machines, and computers realize they have enough IT literacy. The *MIS* course's problems start with the inconsistency between its title and both its content and the direction business is moving. Information systems specifically for managers are playing a less central role as the people who do the work take on more self-management responsibilities and as information systems that help people directly in their work become more prevalent. If this is a fair description, what should we do?

We have a serious, but soluble positioning problem that can be addressed by focusing on two themes: 1) the skill of creating, analyzing, and using information, and 2) knowledge about the ways information technology can support and improve business processes.

The skill of creating, analyzing, and using information is certainly not our sole bailiwick. Accounting, management science, and statistics all claim part of this territory, but in downsizing universities we need to find ways to collaborate with them. We should provide courses that promote genuine skills such as analyzing data, building models, critiquing information and measures of performance, and using abstract concepts in decision making. If these "skills" are beyond many of our students, I wonder why those students are in universities and what they would learn regardless of what we cover.

Knowledge about business processes is not our sole bailiwick either, since marketing production, and finance also pay some attention to business processes. However, use of IT both as infrastructure and as an element of business processes is a cornerstone of TQM, reengineering, and competitiveness initiatives in many firms. Students going into business really do need to understand this, and it is a topic that would otherwise be slighted or ignored if we don't teach it.

Stating in a more direct and controversial way, I think our major emphasis should no longer be IT per se, but rather, "IT-supported business processes." This gives a rationale for explaining much of what we do today, and also a way to identify the material that is of enduring importance. By focusing on IT-supported business processes we would find a core of useful ideas that change slowly compared to the way technology changes. Our courses would seem less like a catalog of terms to be memorized and forgotten, and more like a coherent body of knowledge students can apply now and when they go to work.



## "Integrating Quality Management into the MIS Curriculum"

Barbara B. Denison

### Abstract

Businesses are radically changing their structure and operation to adopt total quality management (TQM) practices. This paper briefly reports the coverage of quality management in the DPMA model curriculum and in the leading systems analysis and design books. A process for integrating quality into the MIS curriculum is presented.

#### Quality Management in Information Systems in Business

Computerworld surveyed 102 top information systems executives and the findings were reported in the October 19, 1992 issue. Executives were asked to rank the importance of activities to the company's business strategy. A scale of 1 to 5 was used with 5 being most important. Total quality management was ranked 4.23, second only to re-engineering. [21] As business processes are re-engineered using TQM, the information systems also need re-engineering.

Information systems departments are involved with total quality management initiatives in a number of ways. One way is providing the information resources to support TQM. The first businesses to emphasize quality were typically involved in health care or manufacturing. The Society for Information Management (SIM) surveyed its 581 members and asked if they had a company wide quality program in place. The industries with the percentage that responded yes to the question were: health, 71%; manufacturing, 66%; transportation and public utilities, 66%; business and legal, 44%; wholesale and retail, 40%; and education, 25%. The average response across all groups surveyed was 58%. [18]

SIM also asked the members the primary goal of their company's quality program. Improved service was rated highest, closely followed by competitive advantage. [18] Participants were asked to identify the primary role of information systems in their quality program. The responses in order were: participant, 49%; facilitator of process changes, 21%; supplier of technology, 15%; and leader of providing information and analysis, 13%. [18]

Information systems departments are applying quality management to improve their own processes and products. An obvious product is software. Software engineering has focused on accuracy, maintainability,

reliability, efficiency, and more recently design of reusable software components. Some public failures of software have increased awareness of the public's dependence on software and the ramifications of errors. For example, in June, 1991, a reported three faulty lines of code out of more than two million instructions set off the chain reaction of three Bell companies using the software for call routing. As a result, an electronic traffic jam was created that paralyzed the phones of more than ten million customers in Los Angeles, Pittsburgh, and Washington, D.C. [26] One metric being used to measure the quality of software is number of bugs, errors or defects per 1000 lines of code.

Information systems departments have found that just measuring errors in lines of code does not insure quality or customer satisfaction. An executive report on quality in Computerworld, January 6, 1992, reported several reasons why the traditional objective of minimizing defects while conforming to pre-established customer specifications isn't sufficient. First, many IS departments have not identified their customers or asked them what constitutes a high quality product or service. Second, the existing quality techniques haven't been adapted to information systems processes. Third, the work environment and culture of information systems may still highlight individual performance rather than teamwork. Last, corporate management may still view information systems departments as providers of technology rather than a strategic business component. [18]

Total quality management will change information systems departments in different ways depending on the company's goals. As an example, Nashua Corporation began adopting Deming's quality philosophy in the late 1970's. [20] But it wasn't until four years ago, that IS became viewed as a major contributor. Nashua's IS department developed a

five-year strategic plan that resulted in decentralization of IS into Nashua's nine divisions and a flattening of the IS organization. Continuous process improvement and improved customer service were major goals. As Nashua flattened their company's organization and pushed decision making to lower levels, the IS department shifted decision making on hardware and software to teams of IS staffers and end users at the local level. [20]

As a second example, Miles, Inc., a pharmaceutical firm, asked their IS department to take a leadership role in implementing quality concepts. After six months of studying methodologies including Deming and Juran, they chose a program and began training their IS employees. Miles eventually formed 26 quality improvement teams. Information systems teams looked at areas including application development, requirements planning, and data center management. They identified problems, barriers to improvement and fixes. As a result of some concrete successes, their TQM program is going company wide and more than 300 employees have been trained. [19]

The information systems department may also provide hardware and software technology plus training and support for technology to facilitate quality management in their company. The most well-known software for support of quality in manufacturing is statistical quality control (SQC) or statistical process control (SPC). There are over 130 suppliers of this software. [17] The use of project management software and groupware to support team development is growing. As decision making is pushed to lower levels and distributed, the need for access to data bases increases. [28] IS departments will be called upon to support TQM in their organizations.

#### Quality Management in the MIS Curriculum

Businesses have publicly called upon academia to learn, teach, do research, and practice total quality management. In "An Open Letter: TQM on the Campus", published in the Harvard Business Review, the Leadership Steering Committee of the Total Quality Forum called upon businesses and universities to form a new partnership and commitment to quality management. [24] Specifically, they urged universities to learn what TQM organizations are doing and form relationships

with local organizations practicing TQM. Second, form a research agenda. Third, evaluate the business curriculum. [24]

Robert Kaplan of the Harvard Business School did a case study of 19 business schools that had been ranked among the top 20. One purpose was to look at the coverage of quality issues in the curriculum. Kaplan found leading businesses well ahead of academic research and teaching. Looking specifically at operations management, Kaplan found 15 of the schools had three or fewer sessions on quality in their introductory operations management course. [14] Kaplan surveyed four operations management journals and found almost no direct coverage of TQM in their 278 articles. [14]

As part of our college's curriculum review for TQM, we researched text materials in the functional areas to introduce quality management in the core courses. We acquired a series of publications sponsored by the American Society for Quality Control (ASQC). ASQC prepared a series of booklets for textbook authors. The booklets contain important concepts and definitions for inclusion in textbooks plus references for authors. The series was developed for introduction to business, accounting and finance, management, marketing, personnel, production/operations, and strategy courses. [5] [9] [10] [11] [12] [13] [29] ASQC did not develop materials for inclusion in a core course in information systems.

For our systems and analysis design courses, we surveyed a dozen systems analysis and design textbooks published since 1989. We checked for references to quality, quality assurance, and/or total quality management in the index and detailed table of contents. Half of the texts had no references in the index to quality including the texts by the following authors: Amadio [1]; Andersen Consulting [7]; Eliason [6]; Merle Martin [22]; Powers, Cheney and Crow [23]; and Shelly, Cashman, and Adamski [27]. Five of the remaining texts did not discuss total quality management. They did however reference quality assurance. Burch [4], Penny Kendall [15], Saldarini [25], Whitten, Bentley and Barlow [30], and Yourdon [31] had one to three page discussions of quality assurance and/or the quality assurance department. Yourdon's comment typified the discussion: quality assurance people "don't get involved in the project until the very end - after the

systems analysis, design and programming work have been finished, and formal testing activity has commenced. At this point, of course, it is very difficult to make major changes to the system." [31]

The second edition of Systems Analysis and Design by Kendall and Kendall has a chapter entitled "Quality Assurance through Software Engineering." [16] They discuss Total Quality Assurance (TQA). This was the only text to discuss quality assurance as a process over the whole systems development life cycle. Our MIS curriculum committee selected this book for use beginning fall quarter, 1992.

The above survey of texts illustrates the current status of quality discussions in MIS curriculum. Certainly there is a concern for quality software and quality output. The structured systems analysis and design methodology, the use of CASE tools, using prototypes to work with users to define requirements, structured walkthroughs, testing, and post-implementation review are in most analysis and design textbooks. Many of the textbooks have a chapter on system controls. What is not present is a theme of quality management. Deming's 14 points, the process of continuous improvement, and the focus on customer service are lacking. In some ways, the coverage is comparable to the old view of quality control as inspection of final product in manufacturing. Even the Kendall and Kendall book which had excellent coverage of total quality assurance (TQA) relegated it to chapter 20 (out of 21). [16]

Our curriculum review committee also studied the Data Processing Management Association (DPMA) Model Curricula for the 1990's [8] since our MIS major was based on the CIS-86 Model Curricula. The eleven underlying principles of IS-90 do not directly address quality management. They do address integration of IS with the organization and support for the organization's strategic goals. They also state that "increasing attention is being directed to problem avoidance. Error control and risk management are areas that the IS professional must understand and apply." [8] The detailed body of knowledge with competency levels does not address quality management.

#### Integrating Quality into the MIS Curriculum

Our college is following the process suggested by the

Leadership Steering Committee of the Total Quality Forum. [24] The first step is education of the college faculty. We are meeting with local organizations that are advanced in the use of TQM. We are fortunate to be in close proximity of a number of leaders in TQM and to have a core group of faculty that have done work in areas such as statistical quality control and TQM in accounting. A college resource center has been established with materials, videotapes, etc. for use of the college faculty. A TQM day is planned to bring in industry representatives to meet with our faculty. The college also has an active board of advisors of community and business leaders.

Our TQM curriculum committee wrestled with the issue of practicing quality management versus teaching it. Can we as a college integrate quality management in the curricula before we practice it as a management philosophy? Can the College of Business practice TQM if the university has not adopted the philosophy? To avoid "analysis paralysis", we have begun the integration of quality management in the curriculum as the college and university are also reviewing its implications. The state has mandated external reviews of each state university's operations, although the state's primary goal is to deliver higher quality education with much fewer resources! The university has established a center for excellence in teaching and has sponsored a number of programs for faculty.

The second recommendation of the Task Force on Quality is for the college to establish a research agenda on total quality management.

The third recommendation is to take an inventory of the curriculum and look at the proportion of quality content in the core and elective courses. A key first step for information systems is to identify what quality means to IS. This involves identifying the customers and what their needs are. We have identified a number of key concepts. We could not integrate quality management into the MIS major and service courses without examining the college curriculum.

One area to be taught is the quality philosophy and its role in business. The college TQM review wrestled with competing gurus: W. Edwards Deming, Philip Crosby, Armand Feigenbaum, Kaoru Ishikawa, and Joseph Juran, etc. [3] If a specific approach is predominant in an area, it may be adopted. We

decided the overriding themes could be extracted and included in the curriculum.

The college as a whole is addressing the excessive compartmentalization of our curriculum. As with many business schools, our courses are separated into functional areas with little or no team teaching. Only the two capstone management courses have projects that cross functional areas: a simulation game and a strategic planning project. These courses consciously arrange teams with a mix of majors. Our MIS majors also take Accounting Systems. This class mixes accounting and MIS majors to get a cross-functional approach to the role of systems in accounting.

How the IS department specifically supports quality management is a second area. Our college integrates a statistical quality control package in the introductory statistics and production sequence. We are incorporating project management tools and CASE tools in the systems analysis and design courses. We chose the Kendall and Kendall text [16] and are introducing quality management earlier in the systems analysis and design course. This is supplemented with outside reading. Richard Zultner specifically adopted Deming's 14 points to MIS management and identified an MIS version of the seven deadly diseases for software quality. [32]

The DPMA IS-90 curriculum has two courses that focus on managerial/organizational needs. IS-2 is an introductory information systems concepts course. The model states to "describe the role of IS in management including current professional practices and methodologies." [8] This course (or an equivalent) is an obvious place to introduce quality management. The textbook can be supplemented with outside readings and speakers.

IS-9, Management of IS, deals specifically with the role of the CIO, and management of the information systems functions. This is a logical place to cover applying TQM methodology to the IS function. The course syllabus includes CIO and staff functions, planning and control, IS policy formulation, change and project management, and implementation and testing strategies. Measurement of quality would be an appropriate topic which is not specifically included. For example, benchmarking could be introduced. Benchmarking has been successfully

applied to study data center costs and operations and allow businesses to gauge their costs, efficiency, and staffing against other organizations. The IS department at Texas Instruments developed what it feels is the best user survey in the nation. The IS quality director is willing to share their survey with other organizations in exchange for information about their successful practices. [2]

IS-8 in the DPMA model curriculum is the systems project with the goal of applying project management techniques with students working on a group project. We assign teams of students to an actual analysis or design project with an outside client. The client is typically a small business or a non-profit organization. Students can practice their analysis and design methodologies as well as gain experience with a team. Structured walkthroughs of design emphasize a team approach to problem solving. With an outside client, customer satisfaction is a very high priority.

## Conclusions

Business has challenged academia to become leaders in quality management. A total quality management approach is not currently incorporated in most IS curricula or textbooks. This paper has presented recommendations on concepts and strategy for incorporating quality management into the information systems curriculum. Adding lectures and reading alone will not be sufficient. The larger issue of integrating quality management into the business core needs to be addressed. Business schools need to look at their research initiatives and their management practices.

## REFERENCES

- [1] Amadio, William. 1989. Systems Development: a Practical Approach. Santa Cruz: Mitchell Publishing, Inc.
- [2] Betts, Mitch. "Benchmarking helps IS improve competitiveness." Computerworld. November 30, 1992. 1,20.
- [3] Bowles, Jerry and Joshua Hammond. 1991. Beyond Quality: How 50 Winning Companies Use Continuous Improvement. New York: G. P. Putnam's Sons.

- [4] Burch, John G. 1992. Systems Analysis, Design, and Implementation. Boston: boyd & fraser publishing company.
- [5] Dubose, Philip B. 1990. Personnel and Quality. Milwaukee: American Society for Quality Control.
- [6] Eliason, Alan L. 1990. Systems Development: Analysis, Design, and Implementation. Glenview, Ill.: Scott, Foresman and Company.
- [7] Flatten, Per O. and Donald J. McCubbrey, P. Declan O'Riordan, Keith Burgess. 1989. Foundations of Business Systems. Orlando: The Dryden Press.
- [8] Information Systems: The DPMA Model Curriculum for a Four Year Undergraduate Degree. 1991. Park Ridge, Illinois: Data Processing Management Association.
- [9] Johnson, Ross and William O. Winchell. 1989. Business and Quality. Milwaukee: American Society for Quality Control.
- [10] Johnson, Ross and William O. Winchell. 1989. Management and Quality. Milwaukee: American Society for Quality Control.
- [11] Johnson, Ross and William O. Winchell. 1989. Marketing and Quality. Milwaukee: American Society for Quality Control.
- [12] Johnson, Ross and William O. Winchell. 1989. Production and Quality. Milwaukee: American Society for Quality Control.
- [13] Johnson, Ross and William O. Winchell. 1989. Strategy and Quality. Milwaukee: American Society for Quality Control.
- [14] Kaplan, Robert S. "The Topic of Quality in Business School Education and Research." Selections. Autumn, 1991. 13-21.
- [15] Kendall, Penny A. 1992. Introduction to Systems Analysis and Design: A Structured Approach. Dubuque: Wm. C. Brown Publishers.
- [16] Kendall, Kenneth E. and Julie E. Kendall. 1992. Systems Analysis and Design. Englewood Cliffs, NJ: Prentice-Hall, Inc.
- [17] Kern, Jill Phelps. "The QIS/TQM Balancing Act." Quality. 1991. 30(12). 18-41.
- [18] Laplante, Alice. "For IS, quality is 'job none'." Computerworld. January 6, 1992. 57-59.
- [19] Laplante, Alice. "Miles building quality from an IS foundation." Computerworld. January 6, 1992. 63.
- [20] Marengi, Catherine. "Nashua keeps quality flame burning in customer service." Computerworld. January 6, 1992. 61.
- [21] Margolis, Nell. "Marching Orders." Computerworld. October 19, 1992. 108.
- [22] Martin, Merle P. 1991. Analysis and Design of Business Information Systems. New York: Macmillan Publishing Company.
- [23] Powers, Michael J., Paul H. Cheney and Galen Crow. 1990. Structured Systems Development: Analysis, Design, Implementation. Boston: boyd & fraser publishing.
- [24] Robinson, James D., John F. Akers, Edwin L. Artzt, Harold A. Poling, Robert W. Galvin, and Paul A. Allaire. "An Open Letter: TQM on Campus." Harvard Business Review. November-December, 1991. 94-95.
- [25] Saldarini, Robert A. 1989. Analysis and Design of Business Information Systems. New York: Macmillan Publishing Company.
- [26] Schwartz, Evan. "Turning Software From a Black Art into a Science." Business Week. Special Issue, October 25, 1991. 80-81.
- [27] Shelly, Gary B., Thomas J. Cashman, Judy Adamski, and Joseph J. Adamski. 1991. Systems Analysis and Design. Boston: boyd & fraser publishing company.
- [28] Stodolak, Frederick and Joseph Carr. "Systems Must Be Compatible with Quality Efforts." Healthcare Financial Management. 1992. 46(6) 72-77.

[29] Suminski, Leonard T. Jr. and Ross Johnson. 1990. Finance, Accounting, and Quality. Milwaukee: American Society for Quality Control.

[30] Whitten, Jeffrey L., Lonnie D. Bentley, and Victor M. Barlow. 1989. Systems Analysis & Design Methods. Boston: Richard D. Irwin, Inc.

[31] Yourdon, Edward. 1989. Modern Structured Analysis. Englewood Cliffs, NJ: Yourdon Press.

[32] Zultner, Richard. "The Deming Approach to Software Quality Engineering." Quality Progress. November, 1988. 58-64.

# INCORPORATION A GIS COMPONENT IN AN INTRODUCTORY COMPUTER SCIENCE COURSE

Carol A. H. Hall, Krishna K. Agarwal, Alfred L. McKinney  
Department of Computer Science & Geography, Louisiana State University in Shreveport, One University Place,  
Shreveport LA, 71115

## ABSTRACT

Geographic Information Systems (GIS) are becoming prevalent in government, commercial and educational applications. GIS systems utilize powerful computer hardware and software to store and retrieve geographical information, automate spatial analysis, manage resources, and predict complex dynamic systems. We present the recent incorporation of basic GIS into our computer literacy course.

## INTRODUCTION

Louisiana State University in Shreveport (LSUS) is located in Northwest Louisiana across the Red River from Barksdale Air Force Base (BAFB). Representatives of LSUS, Barksdale and the City of Shreveport meet regularly to consider common problems and discuss possible solutions. In 1990, it became clear that all were involved in mapping and using maps of the area for various applications (utilities, police, street maintenance, cable, etc.) and that all would be well served if residents of the area were more knowledgeable in the field of computer mapping and the use of Geographic Information Systems (GIS). The problem was presented to the LSUS campus community and ultimately the faculty decided that, wherever possible, we would incorporate this technology into appropriate courses. Additionally, when a geography position became available last year, LSUS sought, found, and hired a geographer with a specialty in Geographic Information Systems.

The cooperation between BAFB and LSUS resulted in the paper listed in [7] and also three GIS contracts which have recently been completed - the projects involved modeling the infrastructure of BAFB plus analysis of their runways.

Recently, LSUS and the City of Shreveport have

entered into a contract to conduct a survey of the area comprising the Interstate 49 corridor within the city limits of Shreveport. The project involves faculty, graduate and undergraduate students.

These activities demonstrate the power and wide applicability of GIS plus the necessity for acquainting our students with the technology.

## METHODOLOGY

Computer Science 111, Introduction to Computers, seemed to be one of the most logical courses to begin the introduction of our students to ideas and concepts of GIS [1]. The course currently includes orientation to the history and use of computers, hardware and software, programming in BASIC, and use of application programs for word processing, database management, and spreadsheets. However, the development of personal computer usage skills does little toward helping the non-scientific student to comprehend the potential of computers to assist in solving problems. At LSUS, we believe that both the climate in undergraduate education and the current state of computing technology support a change in emphasis from personal computing to information analysis. We have begun to create an "intro" course with this goal. To this end, we are now incorporating a GIS component into the course. With careful scheduling of the semester, and a slight

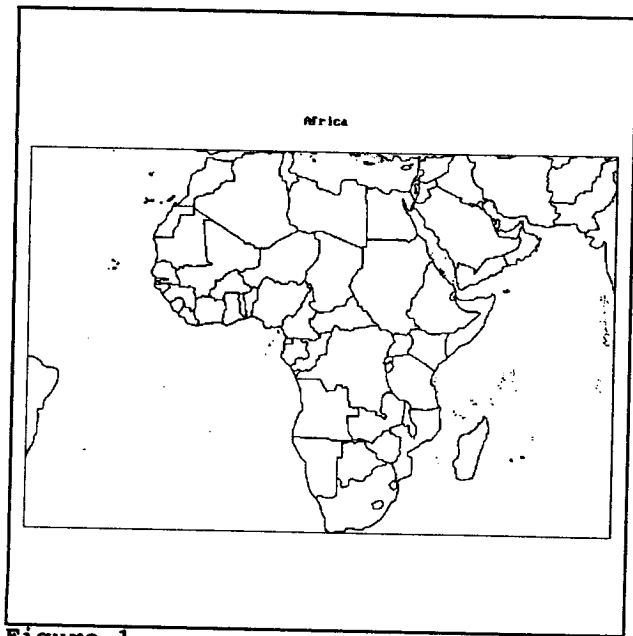


Figure 1

de-emphasis on programming, we have been able to set aside a bit more than a week for the introductory study of GIS. We schedule it to follow the database study since GIS is actually a multiple layer database.

We begin by showing one or two selected video presentations (available from the Environmental Systems Research Institute, Incorporated, Redlands, CA) on the use of GIS in cities and for conservation studies [5, 6]. In cities GIS is used for routing of emergency vehicles, flood control, the design, planning and maintenance of roads, sewerage, electricity, telephone and other utilities. After an introductory discussion, the computer is used in the classroom to demonstrate the concepts of computer mapping and the variety of data that can be represented on such a dynamic medium. We actually use PC Globe Software [3] (inexpensive and readily available at computer and discount stores) and begin with rather simple and small scale maps to illustrate the difference between vector data and raster data [4]. We discuss mapping various types of data such as elevation, climate, cities and population. A number of these are produced in class with the goal of making the software familiar to the student. It does not take long for the student to

want more detailed information. Class discussion usually suggests that if other information could be "layered" onto the maps, the result could provide some interesting maps. Figure 1 shows a map of Africa produced by PC Globe. It has the country boundaries "layered" onto the vector outline map.

We do not have a more sophisticated GIS software package than PC Globe that we consider friendly enough, or available enough for our students to obtain and use to prepare more realistic maps. Therefore, we constructed some maps for presentation using IDRISI software, a professionally oriented GIS software package [2]. Several more complex maps with various "layering" of information were produced and used to foster class discussion. Examples included rainfall layered over agricultural products, temperature layered over landforms, etc.

While there is a limit to the sophistication of the maps that can be produced with PC Globe software, the program has a very useful feature. It will display any file that is in proper PCX format. We use this feature to import the maps created with IDRISI software so we could easily display them to the class.

We were therefore able to discuss some additional and some more complex mapping concepts such as orthographic projections. It also allowed us to use shades of color to enhance the maps, and layer other data onto maps which we displayed in class on a computer with a display panel and an overhead projector. For example, the orthographic map of Africa, was layered with a color map showing annual rainfall. We also combined the orthographic map with a vegetation map, and encouraged class discussion of the observed results. See Figure 2 for an orthographic map of Africa shown from the perspective of 30 degrees above the south pole with scaled elevations.

After the classroom demonstrations and discussion, the computer lab is set up with the PC Globe software and the students are given an



exercise to be done outside of class. The instructor assigns each student a different country to encourage comparisons and discussions among the class members (see Appendix). The exercise requires about an hour or two of the student's time. An informal survey of the students reveals that they have enjoyed working with PC Globe, especially when they have discovered unexpected facts about various countries and cultures that they can use for their other classes.

## CONCLUSION

We are confident that our first step in incorporating GIS in our general education program is a good one. Other departments at LSUS are presenting and using Geographic Information Systems appropriate to their content. We have applied for several grants that would help us to extend our GIS offerings. We further hope that our efforts will serve as a model for other universities to increase their emphasis on this important approach to managing resources and predicting complex and changing systems.

## REFERENCES

- [1] Goodchild, M.F. and Kemp, K.K., Introduction to GIS NCGIA Core Curriculum. National Center for Geographic Analysis, University of California, Santa Barbara, CA, 1990.
- [2] Eastman, J. R., IDRISI, A Grid-Based Geographic Analysis System, Clark Univ. Graduate School of Geography, Worcester, MA, 1990.
- [3] PC Globe, PC Globe, Inc., 4440 S. Rural Rd., Tempe, AZ, 1991.
- [4] **Geo Info Systems**, Aster Publishing Corp., Eugene, OR. Various issues have been made available to the faculty teaching the course.
- [5] **GIS Government's Information Solution**, A Video Presentation, Urban and Regional Information Systems Association, 900 2nd St. N.E. Suite 300, Washington, D.C. 20002.
- [6] Other Short Video Presentations, Environmental Systems Research Institute, Inc., Redlands, CA, USA.
- [7] Mitchell, W. and Kistler, E., An Alliance for Excellence in GIS and Information Integration at Air Force Model Base and LSU in Shreveport, The American Society of Mechanical Engineers, Atlanta, GA, December, 1991.

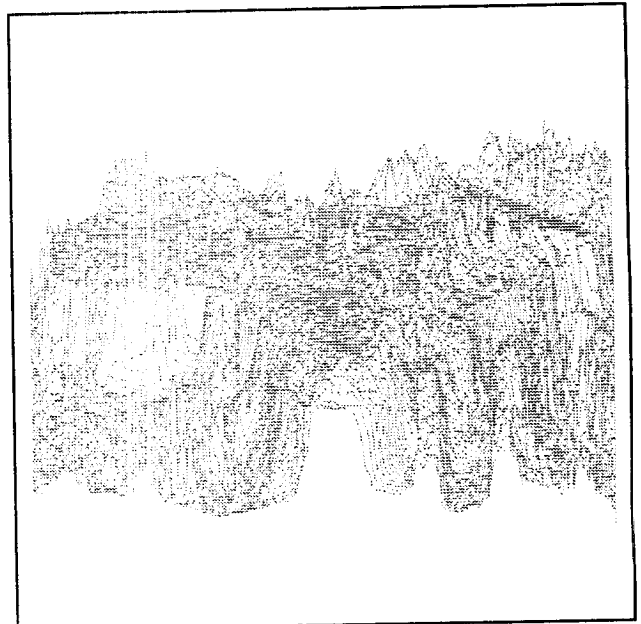


Figure 2

## Appendix - Laboratory Exercise

Country assigned: \_\_\_\_\_

- I. Become familiar with the PCGLOBE environment. Then do the following exercises and write your answers NEATLY in the blanks provided.
  1. Load the program following instructions given by your instructor.
  2. Become familiar with the "Pull-Down" menus by actually accessing each one. Note the selections available on each menu. Complete documentation for PCGLOBE is contained in the HELP menu, so you do not need a manual to use the product. Current prompts are displayed at the bottom of the screen.
- II. Select the World Menu.
  1. Move the cursor to select the world map.
  2. Now that the map of the world is displayed, select Point and Shoot. Notice that a cross-hair cursor appears. You may move this with either the arrow keys or the mouse. Notice that the name of the country being pointed to appears at the top of the screen. Work with this until you are comfortable with how it works.
    - a. Which country is the northernmost body of land shown on the world map?  
\_\_\_\_\_
    - b. Which country is southernmost? \_\_\_\_\_
    - c. Which country is the westernmost in Africa? \_\_\_\_\_
    - d. What body of water is between Australia and New Zealand. \_\_\_\_\_
    - e. Investigate the islands in the Atlantic Ocean. Do any belong to Spain? If so, name them.  
\_\_\_\_\_
  3. Press <Enter> to recall the World Menu. Select Group from the menu. Look over the list of international groups and organizations. Select the Arab League. Now return to the World Menu and select Point and Shoot.
    - a. Which Arab League nation is westernmost? \_\_\_\_\_
    - b. Which one is southernmost? \_\_\_\_\_
    - c. Which one is northernmost? \_\_\_\_\_
    - d. Is Iran a member of this group? \_\_\_\_\_
    - e. List of all those in Africa.
  4. Return to the menu and select the Least Developed countries, and again access Point and Shoot.
    - a. Which country is southernmost? \_\_\_\_\_
    - b. On leaving this country and traveling in any direction, which country(ies) do you enter first?  
\_\_\_\_\_
    - c. Which is the easternmost country? \_\_\_\_\_
    - d. Are any of these in the northern hemisphere? \_\_\_\_\_
    - e. Are any of these countries in the Arab League? \_\_\_\_\_ If so, which?  
\_\_\_\_\_
  5. Now return to the menu and Select Country. All members of your currently selected Group will be highlighted on the country list. Did you get them all? \_\_\_\_\_.

6. Now, while the list of countries is showing, find and select your country. (Your teacher will assign your country-see top of page)
- What countries or other features share a border with your country?  
\_\_\_\_\_
  - What continent is your country on? \_\_\_\_\_
  - Return to the menu and Select Group. Investigate all the group memberships and determine which (if any) of the groups or organizations your country belongs to. List them.  
\_\_\_\_\_
7. Now change to the Country Menu. Your country should be the active country. If it is not, use Select Country and make it so. Now return to the Region Menu and select Active Continent/Region. Note where your country is in this area. Return to the World Menu and note where your country is in the world.
8. Return to the Country Menu. Display a base map of your country.
- What is the population? \_\_\_\_\_
  - What is its area in sq. miles? \_\_\_\_\_
  - What is the highest elevation range? \_\_\_\_\_
9. Select Major Cities.
- What is the capital city of your country? \_\_\_\_\_
  - What is the largest city listed? \_\_\_\_\_
10. Select Elevations to see a topographical map of your country.
- What is its highest elevation? \_\_\_\_\_
  - Is any land under sea level? \_\_\_\_\_
11. Select Features to see the major physical features of your country. What are they? \_\_\_\_\_
12. Select Flag & National Anthem. NOTE: If you are in the lab, be ready to press Esc as soon as the music starts. DO NOT disturb the others in the lab!
- What are the predominant colors in the flag?
  - Sketch the flag here.
13. Now select the Database Menu. Select Data for Active Country. You can view the database screens in sequence by pressing <Enter> to advance to the next screen. To return to the previous screen, press PgUp. Study each screen and supply the following information about your country.
- the projected population for the year 2000? \_\_\_\_\_
  - the population density? \_\_\_\_\_
  - the annual population growth? \_\_\_\_\_
  - the current total male population? \_\_\_\_\_
  - the current total female population? \_\_\_\_\_

- f. the most populous age range? \_\_\_\_\_
- g. the least populous age range? \_\_\_\_\_
- h. percentage of the population in urban centers? \_\_\_\_\_
- i. the major languages \_\_\_\_\_
- j. the major ethnic groups? \_\_\_\_\_
- k. the major religions? \_\_\_\_\_
- l. the birth rate? \_\_\_\_\_
- m. the death rate? \_\_\_\_\_
- n. infant mortality rate? \_\_\_\_\_
- o. number of hospitals? \_\_\_\_\_
- p. number of doctors? \_\_\_\_\_
- q. published literacy rate? \_\_\_\_\_
- r. number of universities? \_\_\_\_\_
- s. the capital city's population? \_\_\_\_\_
- t. capital's latitude-longitude coordinates? \_\_\_\_\_
- u. country's type of government? \_\_\_\_\_
- v. country's current political leader? \_\_\_\_\_
- w. country's GNP in 1989? \_\_\_\_\_
- x. GNP per capita? \_\_\_\_\_
- y. Primary natural resources? \_\_\_\_\_
- z. Primary agricultural products? \_\_\_\_\_
- aa. Primary Industries? \_\_\_\_\_
- bb. Major Imports? \_\_\_\_\_
- cc. Major Exports? \_\_\_\_\_
- dd. Consumption of electricity? \_\_\_\_\_
- ee. Mined or quarried metals or minerals? \_\_\_\_\_
- ff. Annual production of milk? \_\_\_\_\_
- gg. Annual production of rice? \_\_\_\_\_
- hh. annual production of beer? \_\_\_\_\_
- ii. annual production of newsprint? \_\_\_\_\_
- jj. Is a visa required to visit your country? \_\_\_\_\_
- kk. average high temperature in capital in December? \_\_\_\_\_
- ll. average low temperature in capital in June? \_\_\_\_\_
- mm. principal tourist attractions? \_\_\_\_\_

14. Select the Utilities Menu. Investigate the various toggles.
- a. Use the City Distances/ bearings selection to determine the distance between the capital city of your country and New York, NY, USA \_\_\_\_\_ and also between the capital city of your country and London, England. \_\_\_\_\_
  - b. If it is 8am in the capital city of your country, what time is it in Dallas, Tx, USA ? \_\_\_\_\_ and what time is it in Paris, France? \_\_\_\_\_
  - c. What is the main currency in your country? \_\_\_\_\_
  - d. Would you like to visit your country? Why of why not? (answer on back)

# COMPUTERS: DEVELOPING AN INTERDISCIPLINARY WRITING SKILLS TOOL

MARIAN SACKSON AND ILENE SIEGEL-DEUTSCH, PACE UNIVERSITY

## ABSTRACT

Most students at the college and university educational levels need to improve their combination of thinking and writing skills. One approach to confronting this dilemma is writing across the curriculum. A student's ability to articulate thoughts through writing are an important component of the total educational experience. However, many disciplines within the educational process often do not emphasize or reinforce the thinking and writing components.

We are a large private university in a major urban Northeast setting. Our student body is predominantly first generation college attendees, close to 50% are minorities, and they come from public and parochial high schools. The average verbal Scholastic Aptitude Test (SAT) score at our university for Fall 1992 entering freshmen was 409. Performance on this objective test supports the need for a broader based focus on reading, writing, and comprehension skills.

At our university all entering freshmen are required to take a writing skills test. As a result, many students need to participate in a special writing course to compensate for their writing deficiencies. More than half of the mainstream entering freshman class are placed into Reading 100, a remedial course devoted to developing effective writing skills through review of basic grammar, sentence structure, and usage.

Thus, due to these numbers of students attending the special writing classes, the academic administrators of our university realize there are serious problems with student's ability to articulate their thoughts through writing. In recognition of this problem, our university offers a "Writing Across the Curriculum" seminar to all interested faculty members.

With the improvement of writing as our goal, we examined the effect of introducing a computer lab writing component into a college level required introductory computer course (CIS101). It was our objective to encourage and support the successful application of microcomputer technology as a tool to enhance our students' writing skills.

The number of students in the pilot CIS101 course was about twenty-four students. Twelve students comprised the experimental group. As part of the CIS101 course, three person student teams produced a newsletter. The student teams, were introduced to the advanced computer equipment and desktop publishing software. The format of the newsletter, published by the students, was of their own design. The other group of twelve CIS101 students had no intervention (i.e. exposure to specific upgraded computer equipment or the writing across the curriculum experience). At the end of the academic semester a post test, measuring writing skills, was administered to both groups.

The measured difference between the writing skills of the randomly chosen experimental groups at the beginning of the semester and end did not show significant differences. These groups, however, did express positive attitudes toward the experience. They both enjoyed working in a group and learning how to format a newsletter with a word processor. They improved the physical appearance of the newsletter during the experimental period. The experiment was a success, however the success was not totally in the writing skills. Some Communications instructors and researchers stated that significant improvements in writing skills require more than one semester in one class experience.

# Business Process Reengineering

*Donald R. Chand & Robin Cannon*  
*Bentley College*

The objective of this workshop is to share our understanding of what is business process reengineering, what are its underlying concepts and principles, what methods and techniques of reengineering have proven to be successful in practice, what are some of the notable reengineering efforts, what is the appropriate literature on this subject, and how to incorporate this interdisciplinary knowledge in an information systems program?

As the economic and competitive climates change dramatically, so must a company's *modus operandi*. No longer are organizations capable of competing in these tough economic time periods with 5-10% increases in performance. Instead, improvements of 50-100% in productivity are required for the 1990s and beyond. The concept of Business Process Reengineering (BPR) is gaining the attention of corporate executives as a strategy capable of radically improving the performance of their business. As organizations begin to implement reengineering programs, executives need people who are capable of successfully organizing and managing a reengineering effort. Our study has convinced us that there is an interdisciplinary body of knowledge that one needs to undertake a business reengineering project. We believe that this body of knowledge can be wrapped nicely within the academic cloak as an attractive information systems elective for graduate students.

The purpose of the workshop is to share our understanding of:

- ◆ the BPR philosophy
- ◆ the concepts underlying BPR
- ◆ the relationship of BPR with other productivity approaches and trends
- ◆ the tools, techniques and processes supporting BPR practice

- ◆ BPR success and failure factors
- ◆ successful BPR projects
- ◆ possible approaches for instilling BPR capabilities in IS graduates

## Course Outline

- I. Roots and Origin of BPR
- II. BPR , TQM & Continuous Process Improvement
- III. Business Process Reengineering Steps
  - ◆ Organizing for BPR
  - ◆ Selecting Process for Innovation
  - ◆ Modeling the Existing Process
  - ◆ Identifying Change Levers
  - ◆ Developing the Process Vision
  - ◆ Designing New Process
  - ◆ Prototyping New Process
  - ◆ Implementing New Process
- IV. Business Process Reengineering Tools / Methods
- V. Time Frame / Project Schedule
- VI. Success Factors
- VII. Failure Factors
- VII. Examples
  - ◆ Mutual Benefit
  - ◆ Ford
  - ◆ Merck
  - ◆ Cigna
  - ◆ Hallmark
  - ◆ ITT Sheraton
- VIII. Academic Issues
- IX. Bibliography

**HIGH SCHOOL COMPUTER LITERACY  
and  
PERFORMANCE IN INTRODUCTION TO CIS COURSE  
IS CURRICULA TRACK**

By

Craig A. VanLengen, Ed.D. (Presenter)  
College of Business Administration  
Northern Arizona University  
Box 15066  
Flagstaff, AZ 86011-5066  
(602) 523-7392  
vanlengen@nauvax.ucc.nau.edu

and

Jo-Mae B. Maris, DBA  
College of Business Administration  
Northern Arizona University  
Box 15066  
Flagstaff, AZ 86011-5066  
(602) 523-7403  
maris@nauvax.ucc.nau.edu

**ABSTRACT**

Class rank, repeating of course, math level, prior computer experience, and work experience after graduation from high school, for students enrolled in introduction to computer information systems, were compared with student performance as measured by average of examination scores. Prior computer experience did not show a significant relationship with student performance. Significant relationships were found for prior work experience, repeating of course, class rank, and math level.

**INTRODUCTION**

**Purpose**

The purpose of this study was to determine whether high school computer courses were assisting our introductory computer students in their performance and if so do we still need to require students with high school computer literacy to take our introductory computer

course. McClanahan and Holden (1989, 1987) concluded that student computer experience in high school did not provide them with the background required to meet the expectations of faculty in the functional areas of business. Their conclusion was based on questionnaire data and requests from faculty in the functional areas of business. In 1987 they also concluded that the level of computer proficiency

desired by business faculty will continue to increase. This increase in desired proficiency means that the introductory computer course at the college level will still have a purpose. Even if high schools improve their instruction of computer literacy they will be aiming at a moving target that will leave a gap between desired proficiency and actual proficiency.

Copenhaver (1991) surveyed incoming students on computer use and content. The conclusion was that even though many students have received some instruction in programming and applications, they still have deficiencies in content that is required at the college level.

Greer (1986) analyzed the achievement of introductory computer science majors and the amount of high school computer science instruction and emphasis on structured programming in high schools. No significant relationship was found between high school instruction in programming and university examination achievement. The only finding was that students with high school computer science experience had a lower withdrawal rate from university computer science courses. Greer questions the purpose of computer instruction in high school. If it is to provide university preparation it is not achieving its goal. If the purpose is to improve problem solving skills the emphasis of the instruction and the content of the high school computer curriculum may have to be modified.

## **Problem**

At Northern Arizona University we are currently

teaching 600 to 800 students per semester in our introduction to computer information systems course offered in the college of business administration. This course is required of all business majors and minors and can be taken by non-business students as a liberal or general studies course.

The students have a diverse background in computers and academic performance. Many students with high school computer background feel that the course is a waste of their time, because they already know everything about computers. If they do know everything about computers then it would be better for both the student and the university to have the student take other material to assist in rounding out their education. However, previous surveys in colleges of business and studies in the computer science area do not support the students' opinion of their computer proficiency.

To find out the relationship of current high school instruction in computers and performance in a college of business administration introductory computer information systems course the students' computer background would need to be determined and compared with their performance in the course. This would answer the question does high school or other previous computer instruction assist in performance in an introduction to computer information systems course at the college level? Our hypothesis was that previous computer experience would have no influence on student performance in introduction to computer information systems.



## Method of Investigation

The data were gathered during the Fall 1991, Spring 1992, and Fall 1992 semesters. Students were enrolled in both small and large lecture sections. Both the small and large sections were taught by full-time non-adjunct faculty.

Besides their previous computer experience other factors including: class rank, major, math level and strength, writing strength, business strength, ownership of a computer, and amount of work experience after high school.

These factors were evaluated to determine which offered the greatest predictability for successful performance in the introduction to computer information systems course.

## ANALYSIS OF DATA

An analysis of covariance was conducted with the survey data. The first test score was used as the covariate and an average of the remaining test scores during the semester was the dependent variable as shown in Table 1 at the end of the paper. The first test score is a concomitant variable. Prior computer experience did not show any significant relationship with performance as measured by the average test scores. However, significant relationships were found for instructor, class level, repeating the introduction to computer information systems course, math level, and work experience after high school.

## CONCLUSIONS

The results obtained from an analysis of variance failed to reject our hypothesis that previous computer experience would have no influence on student performance in introduction to computer information systems. An examination of the relationship of other independent variables indicate the importance of instructor, math level, repeating the course, class level, and prior work experience. Prior work experience besides providing discipline and motivation for the student to do well at college could also represent maturity level of the student. This is also supported by the significance of class level.

Since computer experience prior to college did not show a significant relationship with performance in introduction to computer information systems it indicates the continued need for the course as part of the core college of business curriculum. Requiring students to have this knowledge prior to college would require more coordination between colleges and high schools.

Since most colleges and universities receive students from a number of high schools coordination might not be possible. Also high schools may have different objectives than colleges of business and may not have the resources to try and meet objectives of both colleges of business and computer science. Even if college and high school computer curriculum objectives could be coordinated as McClanahan and Holden (1989, 1987) concluded it may not be possible for high schools to adequately prepare students for

college level computer work because the proficiency required is constantly changing. The computer field and proficiency levels required of functional business areas are also constantly changing.

### RECOMMENDATIONS

Curriculum development for the introduction to computer information systems must be ongoing to meet the needs of the changes in the computer field and the changing level of computer proficiencies required by functional areas of business. Part of this development effort requires the monitoring of input levels of proficiency that students bring with them so that they are not duplicated and more emphasis can be placed on other important areas. Efforts at coordination with high schools might assist in identifying what content could be successfully performed at the high school level that would fit with their objectives. However, coordination may be difficult for colleges and universities that have students from a number of different high schools.

### REFERENCES

- Copenhaver, D. (1991). High school computer use: The influence on college introductory computer courses. Interface: The Computer Education Quarterly, 13(3), 10-14.
- Greer, J. (1986). High school experience and university achievement in computer science. AEDS Journal, 19, 216-225.
- McClanahan, A. H., & Holden, E. J. (1989). The continuing need for the introductory business computer course. Interface: The Computer Education Quarterly, 11(2), 27-30.
- McClanahan, A., & Holden, E. (1987). Computers in the business curriculum: Is the introductory business computer course a passing fancy? The Journal of Computer Information Systems, 27(4), 23-25.

Table 1

---

General Linear Models Procedure

Dependent Variable: AVGTEST

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	62	223991.8078	3612.7711	36.55	0.0001
Error	1447	143016.0351	98.8362		
Corrected Total	1509	367007.8428			

R-Square	C.V.	Root MSE	AVGTEST Mean
0.610319	10.56917	9.941642	94.0626932

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Test 1	1	121772.2143	121772.2143	1232.06	0.0001
Class Level	5	1234.0734	246.8147	2.50	0.0292
Major	5	233.5197	46.7039	0.47	0.7969
Repeat	1	1342.8056	1342.8056	13.59	0.0002
Math Level	3	2247.3956	749.1319	7.58	0.0001
Word Processing	5	288.9196	57.7839	0.58	0.7118
Spread Sheet	4	325.8295	81.4574	0.82	0.5097
Database	4	523.2427	130.8107	1.32	0.2590
Programming	4	220.2693	55.0673	0.56	0.6938
IBM	4	111.5776	27.8944	0.28	0.8896
Macintosh	4	134.4538	33.6135	0.34	0.8510
Writing Strngth	4	437.5613	109.3903	1.11	0.3517
Math Strength	4	485.0797	121.2699	1.23	0.2974
Bus Strngth	4	411.8092	102.9523	1.04	0.3843
Own Computer	2	189.7875	94.8938	0.96	0.3831
Work Experience	3	2696.3999	898.8000	9.09	0.0001
Instructor	4	7866.1370	1966.5343	19.90	0.0001
Small Section	1	2.8703	2.8703	0.03	0.8647

---

# **The Role of the Introductory Information Systems Course in Recruiting Information Systems Majors**

**Judith C. Simon**

**Ronald B. Wilkes**

Management Information Systems and Decision Sciences Department

Fogelman College of Business and Economics

Memphis State University

Memphis, TN 38152

(901) 678-2462

## **Abstract**

A major issue of concern to faculty teaching in the information systems (IS) area is how to attract students into the major and, more importantly, how to attract *good* students into the major. Students enrolled in an introductory course in IS participated in a study to measure their levels of interest in several IS related job activities at the beginning and end of the course. The research addressed the following questions.

1. Does the introductory IS course cause students to become more (or perhaps less) disposed to selecting IS as a career?
2. Do we attract the better students (those with a higher grade point average) as a result of having taken this course?
3. Does the impact of the course differ by student major?

Results indicate that, in general, the level of interest in IS related job activities decreased by the end of the course. This course in this institution is not causing a majority of students in any major to become more interested in careers in IS. In fact, many students become less interested in performing a variety of specific IS job activities by the end of the course. Students' majors and grade point averages are not, in general, significantly related to changes in levels of interest in IS job activities.

The instrument developed to assess the impact of the course on career orientation will be used to in subsequent research to evaluate the impact of alternative approaches to course delivery. It could serve as a guide for other faculty to evaluate the degree to which their courses meet their objectives.

## Development of An Information Systems Curriculum for Non-traditional Students

James B. Pick and K. D. Schenk

Department of Management and Business  
University of Redlands  
Redlands, California

### ABSTRACT

As the demographic and educational composition of the U.S. work force change, it is necessary to develop curricular programs suitable to older students who may be changing career focus, undertaking a new career, or obtaining education to enter the work force for the first time. This paper focuses on development of, and suggestions for, an undergraduate information systems curriculum that is suitable to such non-traditional, adult students.

### 1. INTRODUCTION

The commercial information systems (IS) field is now about 40 years old. During that period, the education and training of information systems professionals has been shared between higher education and corporate training/educational programs. Many people who have entered information systems completed formal education in other areas and were drawn into the computing field through the ever-increasing demands for workers. Many of the people drawn in were adult workers. Most of the higher education degree programs in information systems focus on younger students in the traditional age range of 18 to 25 years old.

More recently, some information systems undergraduate programs have been designed to serve the needs of the non-traditional, adult student (Schrage and Schultheis, 1984-85). There are several factors leading to increasing demands for such programs. First of all, the U.S. work force has been growing older (International Labor Office, 1986; Johnston, 1991). The I.L.O. estimates that the age of American workers will average 38 1/2 years in 2000, versus 36 years in 1980. At the same time, baby boom population cohorts have experienced job squeezes, which have led many in this overcrowded generation to make shifts or outright changes in their career paths. Another factor leading to increased career path changes is the substantial economic recession that has gripped the U.S. since 1991, with current mixed signals concerning recovery. The recession has resulted in "downsizing" of the labor force and an increase in layoffs and unemployment (Greenwald, 1992). The frequency of career changes was consistently high in the 70s and 80s. In the 1980s ten percent of the American work force changed careers each year (Markey and Parks, 1989). These career shifts often involved movement into technology fields, including information systems.

At the same time, the number of information systems jobs is forecast to grow substantially in the 1990s. For instance, the U.S. Department of Labor (1990) study analyzed the top 50 fastest growing jobs for the period 1988-2000. Seven of these top jobs were in information systems. More importantly, IS jobs accounted for 21 percent of the projected numerical growth among these fifty job categories, a total of 1.2 million new jobs over this period.

In the United States, there is a long history of adult education programs to provide education for older, non-traditional

students. The formal concept of adult education stems from 1935, as part of the first decade of the American Association for Adult Education. Following the second world war, there was major growth in adult education concepts and theory, as well as practice (Houle, 1992). In the 1970s, the concepts of lifelong learning appeared in Europe, especially in France, and moved to the U.S. Recently, a focus has developed on global lifelong learning with comparison among different nations and cultures (Thomas and Ploman, 1986; Bhola, 1989). In the U.S., recent academic study has focused on differences in multicultural societies in forms of adult education (Cassara, 1990). The perspective of multicultural education is important for American adult education in the late 20th century, since ethnic minority groups overrepresent those seeking adult undergraduate education compared to the composition of traditional 4-year undergraduate programs.

Global competition has highlighted the need for U.S. workers to increase their educational levels in order to compete with highly educated Western European and Japanese work forces. This need has been emphasized in proposals from the Clinton Administration, under the rubric of job retraining (Dentzer, 1993; Pierce, 1993). The information systems field is a focal point for proposals to create a more educated work force, since there is increasing recognition of electronic technology in global business competition.

In summary, the U.S. work force has shifted in composition and structure to accentuate the need for adult degree programs in information systems. Through case study and suggestions, this paper seeks to inform, assist, and encourage information systems academic programs to develop adult undergraduate programs to serve the clear need of increasing numbers of non-traditional students.

### 2. DEVELOPMENT OF AN ADULT INFORMATION SYSTEMS CURRICULUM AT WHITEHEAD CENTER, UNIVERSITY OF REDLANDS

#### Background

The Whitehead Center is part of the University of Redlands, a private university of about 4,000 students. The Center has the focus of providing undergraduate and graduate degree programs to adult students located in southern California. The center currently enrolls 2,200 students, on a private university campus of about 4,000 students. Within the Center, there are four major programs: undergraduate business administration, undergraduate information systems

(IS), an MBA program, and a small education program (undergraduate and graduate). The university as a whole was founded in 1907 and has had a liberal arts emphasis over its entire history. This emphasis is also reflected in Whitehead Center programs. The university includes as part of its mission statement an emphasis on liberal arts. This emphasis permeates all degree programs, including business.

In the Whitehead Center, classes are provided mostly on campus and at four regional centers, which are located in Woodland Hills, Los Angeles, Irvine, and San Diego, between 60 and 110 miles from the main campus. In addition, some other classes are offered at smaller "satellite locations." The Whitehead Center was founded in 1976, with initial classes offered only on the main campus. Since its founding, it has increasingly responded to the need for regionalization in southern California. This refers to the distribution of instruction, support staff, and facilities to off-campus locations.

The driving factor in regionalization has been the need for adult students in degree programs to have classes located near their workplace, particularly in traffic-congested southern California. As regionalization has taken place in teaching there have been concomitant needs for services and facilities located regionally to support instruction. These resources include admissions and testing personnel, regional center administration, computing and library facilities, and staff. Currently, the regionalization process is locating full-time faculty permanently in the four regional centers, and has added local area networks for instructional computing and library access.

Another aspect of the Whitehead Center programs is a substantial adjunct faculty, many of whom have doctorates and teaching experience, but are working full time in industry. These faculty members contribute a wealth of practical knowledge and credibility to classroom instruction. The working adult student can relate to, and is often enthusiastic about, the practical concepts and emphases of adjunct faculty who are corporate based.

The Information Systems Program in the Whitehead Center  
The Information Systems Program in the Whitehead Center offered its first undergraduate degree program in 1986. It has graduated an average of about 60 students per year since 1988, and currently has about 175 students enrolled.

The degree program requires a total of 120 units, in which students enter with about half the units, mostly consisting of general education courses taken at other universities or colleges. The student completes a required sequence of courses that include information systems courses, business courses, and general education electives.

The program is given in an accelerated format. Specifically, accelerated refers to concentrated weekly class sessions (4 hours on average), short course length (mostly 6 weeks), and a somewhat higher ratio of study time to seat time (i.e. in-class time) compared to traditional programs. It is important to note that the course unit, defined as 40 hours study time plus seat time, remains the same for accelerated and traditional formats. The accelerated format's increased amount of study time per week is quite challenging to students who work full time in corporations. It underscores the students' high level of commitment to their education.

A unique feature that has been present from the beginning of the Whitehead Center is the cluster group concept. For the two years that the student is enrolled at University of Redlands, a group of about 15 students (cluster) enters the program together and stays together for the entire two-year course sequence. This approach has several advantages. In particular, it enables the program to offer the student a very stable schedule and location for his/her college experience. Generally, the student meets in the same location throughout the entire program and has the same weekly class schedule for the two years. This stable location and scheduling is highly advantageous to the adult working student, especially with the commuting and traffic problems of a huge metropolitan area like southern California. Another plus to the cluster approach is that the group often develops its own cohesiveness and interaction patterns, which provide the student with a comfortable and stable behavioral environment for learning. The analogy would be a doctoral or post-doctoral research group, which often provides a rich behavioral base for learning. The disadvantages of the cluster group are: lack of stimulation from new ideas and concepts of changing groups of students, and problems logistically in varying courses, since each student cluster proceeds through the same course sequence.

One advantage of the cluster group approach for information systems programs is the encouragement it can give to interpersonal communication and class discussions. Since the group members know each other well they tend to communicate and interact with each other more. The cluster provides a safe environment in which to practice their interpersonal skills. Many IS studies have called for the curricular need for interpersonal communication (Larabee, 1992; Nantz, 1992; Vollack, 1990; Sumner, 1990). The cluster generally has a group dynamic that encourages communication.

Synopsis of the Whitehead IS Program 1993 curriculum  
The IS Program at University of Redlands revised its curriculum in 1992. This process will be described, along with the resulting curriculum, many of the issues involved, and how they were resolved. First, a synopsis of the resulting curriculum is presented.

The curriculum is 60 units, and is equivalent to the third and fourth years of a traditional undergraduate program. Thus, students enter in the third year, since they have completed the equivalent of the first two years at other universities or colleges. As seen in Table 1, the required courses consist of ten IS courses and eight courses in business and management. In the third and fourth years students also typically take two to ten elective courses in general education. The number of general education courses depends on credit and breadth requirements.

The prerequisite course requirements of Introd. to IS and PASCAL Programming can be met by taking them through the University of Redlands, through another college, or by challenge examination. It is important that prerequisites for non-traditional students offer alternatives. Some entering students have been IS senior managers in major corporations with over 20 years of experience and have never taken a computer or IS course. There needs to be a means for such students to test their computing experience, i.e., by challenge. In fact, university policies allow a student to challenge by exam any course on campus, and some "senior IS manager" students have successfully challenged not only

the prerequisites but also two or three major courses. By contrast, other students enter the program seeking a radical career change from jobs unrelated to computing, and have very little computing background. Those students will take the full prerequisite sequence offered by the University of Redlands. It is important for a non-traditional IS curriculum to offer a range of alternatives for entry, since such a program tends to attract a spectrum of computing experience, knowledge, and skills.

The IS part of the curriculum has a "generalist" focus, i.e., it does not emphasize a particular track or specialty. This "generalist" content corresponds well with the DPMA-90 Model Curriculum discussed below. There are several aspects of the curriculum that justify special attention:

1. Computer language. The student meets this requirement with an introductory third generation language, which can be PASCAL, COBOL, C, or another language. The requirement can be met by taking the University of Redlands' PASCAL course, transfer of an acceptable transcribed university course, or by challenge examination. Every student takes a required course in COBOL, which builds on the structured programming requirement. COBOL was selected for the required course because studies indicate that it is the leader among third generation languages as the language sought in the job market (Athey and Plotnicki, 1991; Litecky and Arnett, 1992). The goal in COBOL is to provide an intermediate depth of knowledge.
2. Student software development project. In his/her final year in the program, the student undertakes a 9-month software development project in a real-world organization, which involves following carefully prescribed and structured steps in systems development. The project usually involves writing software in a third or fourth generation language, although occasionally it involves substantial modifications to existing software packages. The student works under the supervision of an IS faculty advisor, as well as a "site contact" (an employee at the organization willing to evaluate and sign off on project deliverables). The software project has been a staple in the IS Program since it began. In a 1992 survey, IS students considered it to be the most important learning aspect of the program. The project is essential because it offers the student insight into the practical world of information systems, and serves as a controlled "practice run" in the real world to learn or improve systems development skills.
3. Emphasis on interpersonal communications skills. The IS Program stresses the development of interpersonal communications skills. These have been emphasized consistently as crucial skills for IS graduates and IS employees (Couger and Zawacki, 1974; Mackowiak, 1991; DPMA, 1991). For instance, in evaluating IS job listings in well-known job guides, Mackowiak identified communication/verbal skills as the most important skill sought (higher than computing and quantitative skills or work experience). The new IS curriculum encourages these communication skills through discussion emphasis, course standards, and small group interactions.

As a concluding activity in the curriculum development, a 400-page IS Program Faculty Teaching Manual was produced. This manual, provided to all IS faculty, includes a model syllabus and faculty training manual for each IS course offered, as well as recommended instructional computing labs. All of these materials emphasize interpersonal communications through student presentations, class discussions, small group discussions, a variety of approaches

to case studies, etc. The materials encourage faculty members to bring out the full experience of the student body through oral discussions. This strategy is particularly beneficial for adult working students, since their corporate experiences provide them with a wealth of real-world problems and situations to discuss.

#### The Process of Curriculum Development

The new IS curriculum for the Whitehead Center was developed through a collaborative and interactive process with faculty, students, and outside corporations. The process started in mid-1991 with written surveys to all IS faculty and an eight percent sample of the IS student body. The written surveys asked a series of questions in which students and faculty evaluated and rated all aspects of the program and curriculum then in place. It queried them on what improvements they would like in courses, curriculum, instructional computing, applied technical tools and skills, etc. The survey results were surprising, indicating satisfaction with the teaching, the program's real-world emphasis, and the relevancy of the software project, but severe concerns with the technical currency of the program, course sequencing, and the weak or unclear program prerequisites.

The next step was to review information systems curricula and model curricula nationwide, and to interview about a dozen information systems directors in both traditional and non-traditional IS programs. Then the "IS Program Curriculum Review Committee" was formed, which consisted of the Director of the IS Program at U. of R., four U. of R. adjunct IS faculty, one U. of R. management faculty (from a traditional 4-year program), two information systems faculty from Claremont Graduate School, a U. of R. assistant dean of admissions, and two high level IS managers from large southern California corporations. The committee was charged with developing a full curriculum proposal over a six-month period. The committee proved a fruitful setting for, and springboard to, curricular development. The industry association of about half of the members (the adjunct faculty all worked in IS corporate positions) was an especially important factor, since developing curricula for adult working students requires a practical emphasis. The high level corporate IS executives made the valuable contribution of providing vision on the IS strategic directional shifts in industry. For instance, one executive consistently stressed the importance of interpersonal communications and business skills for the IS employee, a viewpoint which reinforced the communications emphasis and substantive business segment of the curriculum. The mixture of viewpoints on the committee was very important in assuring that academic, industry/real-world, and computer skill concerns were all properly addressed.

The 1992 Curriculum Proposal (University of Redlands, 1992) was finished and approved in May of 1992. From then until October of 1992, a group of seven full-time and adjunct IS faculty performed detailed development of the individual courses in the curriculum, resulting in the IS Program Faculty Training Handbook mentioned earlier. Another aspect of implementing the curriculum was a series of IS Program Faculty Training Seminars, five of which were held from October 1992 through March of 1993. Each full-day Saturday seminar provided training for two or three of the new IS courses. The seminars were taught by IS faculty and focused mainly on teaching rather than course content. The seminars were an overwhelming success, and will be

continued in 1993-94.

There were other, more administrative concerns, in implementing a non-traditional IS curriculum. These include implementation of instructional local area networks supporting IS software, the establishment of campus support for the student home computer requirement, building new administrative mechanisms for providing software to students, and communicating the changes in focus of the new curriculum to students in a corporate marketplace.

Curriculum development was recognized as an ongoing process by forming a Corporate Advisory Committee to the IS Program. This committee provides planning and policy advice on the program and curriculum, as well as serving as a linkage with the corporate community. The U. of R. corporate advisory committee consists of several IS executives from profit and non-profit organizations, corporate and academic-based faculty, student representatives, and administrative deans. Examples of issues discussed by the corporate advisory committee ranges from the ideal background for prospective students to the need for nurturing creativity within the program. Concerns about addressing larger corporate issues such as downsizing, employee empowerment, and customer responsiveness are all issues that the committee feels are areas of increasing importance in today's world. In addition, to keep the IS undergraduate curriculum abreast of industry knowledge and technology trends, the committee will greatly facilitate maintaining future currency in the curriculum, as well as keeping the program and curriculum well rounded and suitable to a rapidly changing corporate environment.

Correspondence of the new IS Curriculum to the DPMA-90 Curriculum

The following table compares the IS courses in the new IS curriculum with the DPMA-90 model curriculum.

<u>DPMA-90 Core Curriculum</u>	<u>U. of R. IS Curriculum 93</u>
IS-1 Fundamental Concepts IS-2 IS Concepts IS-3 Computer Concepts	Introduction to IS
IS-4 Application Development	Programming language prerequisite: COBOL Programming Techniques
IS-5 Application Design and Implementation	no equivalent
IS-6 Systems Development I IS-7 Systems Development II	Systems Analysis and Design
IS-8 Systems Project	Applied Software Development Project I,II
IS-9 Management of IS	Management and Decision Systems
	<u>other:</u> Database Concepts Telecommunications Computer Ethics

The differences with DPMA-90 in the introductory segment are perhaps less surprising when the spectrum of students' backgrounds entering an adult IS program are considered. In a traditional 4-year undergraduate program, students are

unlikely to be educated in the conceptual knowledge of information and computer technology. This low conceptual level for entering students justifies the DPMA recommending three introductory courses. However, on the average, matriculating adult working students have higher conceptual backgrounds in IS technology, and most fulfill the contents of DPMA's IS-1, IS-2, and IS-3. It is appropriate in a non-traditional curriculum to consolidate the DPMA courses into a single course.

Another difference between the DPMA-90 and U. of R. Curriculum 93 is the seemingly greater emphasis in DPMA-90 on systems development. However, the U. of R. systems analysis and design course is the longest IS course in the curriculum, nearly twice the usual course length, hence providing the same strong weighting on systems development. The lack of a data-base course in DPMA-90 results in that curriculum incorporating substantial coverage of data-base applications into the Systems Development courses IS-6 and IS-7. Many reviews of the DPMA-90 curriculum regard these courses as combining systems development and data-base (see for instance Yellin and Richards, 1992, which refers to IS-6 as "single user data base development"). The U. of R. curriculum maintains the traditional distinction between systems development and data-base rather than merging them. One reason for this has to do with pedagogy and materials. The preponderance of texts do not merge the two together: also, separating the courses allows the students to concentrate on learning one major software package at a time (specifically, dBASE IV and Excelerator) rather than two together, which might overwhelm adult working students. However, each course integrates aspects of the other into it and faculty emphasize the commonality of concepts across both courses.

Although the Telecommunications course is only an elective for DPMA-90, the Redlands IS curriculum development committee felt very strongly that it should be required. The corporate, real-world emphasis of the faculty and students in the program made this decision inevitable, since they are experiencing substantial growth in the importance of telecommunications. Increasingly, job tasks demanding knowledge of telecommunications in the real world are cited by faculty, students, and industry leaders alike. A national survey on the DPMA-90 curriculum of eighty IS departments, nearly all traditional, confirmed a broad consensus on raising the importance of Telecommunications in curricula (Yellen and Richards, 1992). In fact, Telecommunications was tied for fourth place among all DPMA courses in prevalence of sections and was ranked about mid-way in importance among the list of required DPMA-90 courses.

3. ANOTHER ADULT-ORIENTED PROGRAM

Obviously, the U. of R. program discussed above does not represent the only format available for addressing the unique features of non-traditional students. Southern Illinois University at Edwardsville (SIUE) is an example of a different approach to providing education to fully-employed students. In the case of Southern Illinois, the challenge was to develop a program to meet the needs of a nearby military airbase. Since the military employees had a high probability of being transferred on a relatively frequent basis, the program had to be offered in several locations (Schrage and Schultheis, 1984-85). This way the students could be within reasonable flying range even when transferred to another



location. Currently, SIUE provides business programs at eight locations for the United States Air Force Military Airlift Command (MAC), with these programs servicing the surrounding communities as well. Thus, over time, the student population has become both military and non-military.

The format for the SIUE business program is predominantly an intensive weekend format consisting of registration and two weekends of direct instruction (each weekend is 2 1/2 days in length). However, SIUE views their program as consisting of five time periods: registration, the period between registration and the first weekend of direct instruction (first pre weekend), the first weekend of direct instruction, the period between the first and second weekend of direct instruction (second pre weekend), and the second weekend of direct instruction. The final four-hour segment of each of the instruction periods is set aside for review and study, culminating in a midterm or final examination. The pre weekends are three weeks in length and assignments are geared for this time frame. The three-week period between direct instruction weekends allows ample opportunity for extensive homework assignments, academic research activities, and/or individual field projects.

Steerage and Schulteis discuss some of the concerns expressed by the academic community with non-traditional programs such as the one at SIUE. They contend that an equal or greater variety of instructional methods are utilized in this program. In addition to typical lecture and discussion formats, individual and team casework, role playing, business visits and observations, student demonstrations of technology are not only possible but frequently used instructional methods. Other concerns of academic rigor, thoroughness, faculty, and scholarly activities are addressed by Steerage and Schulteis and appear to be without merit. As a final test of equivalence between their non-traditional and traditional programs, studies comparing on-campus and off-campus student achievements were conducted. The studies involved the same faculty, syllabi, and examinations. Comparisons of grades, research projects, and attrition rates over a three-year period showed no significant difference in on-campus and off-campus student performance.

This example highlights both the possibility and feasibility that non-traditional programs can maintain the depth of study and academic rigor afforded by programs offered in a more traditional format. Proponents of non-traditional programs suggest that criticisms surrounding these programs are often a reaction to change, not rooted in statistical data. Although the authors do not foresee this debate ending soon, examples like SIUE suggest that with "careful planning and the appropriate use of instructional methodology" (Steerage and Schultheis, 1984-85, pg. 33), programs for adult students can maintain good academic standards while addressing the needs of a non-traditional clientele.

The issue of program quality is not the format or delivery system. Traditional and non-traditional formats are not a dichotomy. Program structures range from intensive weekend formats such as SIUE to the U. of R. format, with a somewhat accelerated format of 6-10 week courses, to very traditional semester programs. The focus should be less on how the program is structured and more on whether the structure and content fit the student population it serves.

#### 4. SUGGESTIONS FOR THE SUCCESSFUL DEVELOPMENT OF INFORMATION SYSTEMS CURRICULA FOR NON-TRADITIONAL STUDENTS

Although the authors believe that non-traditional programs can provide quality academic education to adult students, they feel that minor modifications to traditional programs do not meet the unique needs of adult students. Non-traditional programs require basic re-engineering of instructional methods and curricula to address fully their educational goals. Given existing literature and available case studies, some guidelines can be developed for starting this re-engineering process.

Through the process of creating the B.S. IS program at U. of R., we present the following suggestions for successful development and implementation of IS curricula for non-traditional students:

##### Include All Stakeholders

In re-designing the IS program, successful curriculum development and implementation must encompass all of the stakeholders in the process. The philosophy of the IS program at U. of R. is to offer a "generalist" IS degree, with a practical, real-world emphasis. Additionally, the program includes a basic core knowledge of business administration, with a focus on interpersonal communication skills.

Since IS is an applied discipline, a close association among faculty, industry leaders, and students is necessary to provide the integration of theory and practice deemed optimal for IS programs. Additionally, the knowledge base within IS is quickly dated due to rapid changes in the field. This requires open communication among academic and corporate-based faculty, as well as industry experts, to stay abreast of this ever increasing and changing knowledge base. There has been much discussion about the need and value of close ties between industry and academia in the development and implementation of IS programs, and yet few institutions actually have them (Jackson, 1991; Longenecker and Feinstein, 1991; Wiersba, 1992). Allowing input from each group of stakeholders is necessary to achieve the quality and focus of a program that addresses both the conceptual underpinnings and the practical emphasis desired.

First, faculty are in a position to provide a meta-view of the discipline and overall educational goals. Davis (1992) indicates that academic leadership is tasked with "identifying and explaining the curricular significance of changes in information management" and to "generalize industry practice and expertise in an ongoing revision of a body of teachable information management concepts, principles, and methods" (pg. 7). Faculty provide the pedagogical perspective, such as the quality evaluations of instructional materials. Also, faculty, as well as students, determine the quantity of materials that can and should be addressed both inside and outside of the classroom setting.

Second, students, particularly adult students, are very adept at recognizing the need to connect what goes on in the classroom with their current and future work experiences. They are uniquely positioned to see gaps in their IS knowledge as applied to their work environment.

Last, industry input, both from the public and private sectors, is invaluable. The pace of change in information technology

and its role in organizations is staggering. Information systems as an academic discipline must continuously adjust to these changing technology and business issues (Davis, 1992). Industry leaders and experts are often a vital source of information to allow academia to keep abreast of these technology-based transformations. Additionally, industry remains the primary employer for the graduates of IS degrees. Therefore, industry executives are in a unique position to see the strengths and weaknesses of various programs. Their advice should not be taken lightly. Many executives are thinking not only of their immediate labor force but also project what employees' skill sets must look like to meet the challenges of the twenty-first century.

Conjointly, business must be encouraged to cooperate more closely with IS programs by providing money and opportunities for faculty and student access to business issues and technology applications (Wiersa, 1992). Such cooperation fosters realistic perceptions of daily business practices and issues within a variety of work environments.

#### Curriculum and Program Advisory Committees

One strategy for acquiring input from various stakeholder groups is to include members of each group in both development and on-going program advisory committees. Assessing the strengths and levels of expertise of various members helps to determine which individuals are more appropriate for initial development or for on-going advisory committees.

One method for making sure that different perspectives are included in the decision process is to rotate the members of the on-going advisory committee every couple of years. Using a staggered basis for rotation allows the committee to have fresh perspectives as well as continuity at any point in time.

#### Involvement of Corporate-based Adjunct Faculty

The inclusion of faculty in the development and implementation phases of IS program curricula refers to both academic and corporate-based faculty. Corporate-based faculty, like academic faculty, must meet appropriate graduate level academic qualifications. Utilizing both academic and corporate-based faculty supports the overall philosophy of the program. This combination faculty represents the mix of theory and practice that corresponds to the carefully designed curriculum and helps to maintain its balance. Corporate-based, usually adjunct, faculty bring a unique perspective to IS curricula. They provide a bridge between the academically based faculty and industry leaders. They understand the logistics of beginning and maintaining a quality academic program, while also recognizing industry needs for particular skills. They are often working with cutting-edge technology within their corporate organization and bring a wealth of specialized expertise to the program. Particularly in the IS field, where advances in technology and process changes occur at an ever-increasing pace, having access to individuals at the forefront of these changes is quite advantageous. With appropriate methods for sharing that expertise among the academic faculty, the students, and other corporate-based faculty, the knowledge base of the resulting program can be quite comprehensive.

Along with the strengths that corporate-based faculty bring to an IS program, their weaknesses must also be realized and addressed. Academic faculty may underestimate the continual

training and enculturation needed to give corporate-based faculty the teaching and conceptual skills upon which academics focus by virtue of full-time employment in academia. Just as good IS academics continually canvas technology changes and implications, good corporate-based faculty must continually improve their academic skills. It is incumbent upon the academic faculty to provide the corporate-based faculty with these opportunities.

One strategy that is working within the U. of R. IS program is the on-going faculty training sessions mentioned earlier. Corporate-based faculty are encouraged and/or required to attend training sessions in those courses for which they want certification to teach. These training sessions provide an opportunity for several types of learning to occur. First, they familiarize the faculty with the course materials, the conceptual foundations for the development of particular materials, and which knowledge building blocks are essential for future courses. Second, they allow for all faculty to engage in the exchange of teaching strategies, various perspectives for integrating theory and practice, and for discussions of student issues and concerns involving the course, the materials, or the overall program. Third, they allow the faculty to network with each other and disseminate their various expertises throughout the entire group.

The corporate-based faculty at the University of Redlands also has a wide range of academic skills as well as varied corporate experience. The faculty consists of individuals with many years of corporate experience and related master's degrees (MBA, MS in Computer Science, etc.) to those pursuing or possessing Ph.Ds in Information Systems. This variety of academic qualifications, along with a range of industry experiences, helps blend and strengthen the combination of theory and practice presented to the students. Some academic programs criticize the inclusion of corporate-based faculty. We feel that, with close interaction and continual learning on the part of both academic and corporate-based faculty, the resulting faculty has a strength and quality which far exceeds the sum of the individual members.

#### Managing the Spectrum of Skills and Experiences of Adult Students

In providing programs for adult students, educators often focus on the time and logistic constraints of this group without recognizing the broad spectrum of skills and experiences that may be found in a single classroom. Although all students are required to have a minimum of five years full-time work experience for admission to the IS program, students in a single class may have from five to twenty-five years of experience. In addition to time in the work force, students enter the program with a variety of expertise. Students may be extremely knowledgeable within a very narrow IS focus (i.e., Novell networks, project management of financial systems, etc.) but may be novices in other IS areas.

Structuring courses and materials to address the needs and experience levels of all students can sometimes seem a daunting task. However, there are several strategies for dealing with this issue of diversity. First, the use of pre-requisites helps the faculty to concentrate on particular course content without having to revisit basic IS concepts. Pre-requisites allow those students who may not have the same IS knowledge base as the majority of entering students to get on an even starting plane. To attain minimum standards for

entering students, the IS program's two prerequisite admission requirements can be met in alternative ways. By introducing prerequisite courses to the IS program, the faculty can be certain that students possess certain basic levels of knowledge when entering the program.

Second, students with many years of experience or expertise can be valuable assets in helping novice students to make greater knowledge strides than might be possible with the faculty member alone. Developing student teams with both novice and experienced students, where the experienced members of the team facilitate the rate and depth of the learning experience, can be beneficial for all involved. The novice gets greater individual attention than can be provided by the faculty alone, and the experienced students strengthen their knowledge through their role as team facilitators. These student teams also provide greater opportunities for strengthening communication and interpersonal skills. A multitude of surveys indicate that communication and interpersonal skills are among the most important and desirable skills identified by employers (Laribee, 1992; Nantz, 1992; Pollack, 1990; Sumner, 1990). IS students seem to be particularly lacking in these critical skills (Rifkin, 1987; Nagarajan, 1989). Providing multiple situations for students to practice and enhance these skills in a secure environment strengthens the learning process.

There are two additional ways in which experienced students can contribute to the classroom environment. They can enhance both their own and their classmates' knowledge in the role of "in-house expert," often surpassing the instructor's level of knowledge about a particular topic. Likewise, they can participate in classroom instruction for specific topics. These activities enhance their presentation skills and may provide their classmates with knowledge of cutting-edge technology.

Third, for students with extensive expertise and years of experience in particular subject areas, challenge exams of major courses allow them to forego some basic program requirements and seek additional knowledge through elective courses. Challenge exams allow those students with exceptional backgrounds in a particular IS subject to demonstrate and receive credit for their wealth of specialized knowledge. One IS student with more than fifteen years of IS work experience chose this strategy. This student demonstrated acceptable IS knowledge in three courses by challenge exams and was then able to further broaden his/her background through IS elective courses.

Thus, although adult students with years of work experience may vary extensively in both the depth and breadth of their knowledge base, both use and management of this diversity is possible. Indeed, diversity in the classroom enhances the learning experience. This diversity need not hinder those students on either end of the knowledge spectrum. However, these factors highlight the necessity for re-engineering traditional programs to meet the needs and unique features of teaching non-traditional students. Merely adapting programs designed for traditional university settings is not adequate for a non-traditional population.

#### SUMMARY

Many universities and colleges are recognizing the need and potential for providing quality academic programs for non-

traditional students. Although adult education has existed for decades, recent shifts in the composition and structure of the U.S. work force accentuates the importance of these programs for competing in a global economy. With the steep increase in demand for information systems professionals, IS programs have both an obligation and opportunity to make sure that future information systems workers are highly qualified, regardless of the age at which they receive their education.

This paper presents one strategy for developing and implementing an IS program at the University of Redlands. The most important aspect of this strategy is the commitment to a re-engineering perspective. Program development was undertaken, not with an attitude of minor adjustments to an existing program but with a fresh look at what is required for a quality IS program. Incorporating the input of faculty, students, and industry leaders is a key element in this development process. Each stakeholder group has a unique and valuable perspective for designing a program that integrates both the theory and practice necessary for an IS program.

Recognizing the variability in both skill and experience among non-traditional students, as well as the differences between traditional and non-traditional students, is a necessity. This awareness encourages flexibility and creativity in addressing a more variable student population. Following a popular philosophy in industry today, "customization" to meet the unique needs of a changing student population is important for producing a successful academic curriculum. This means that all IS programs, even all IS programs for non-traditional students, should not look the same. A common foundation of conceptual and practical skills provides some consistency and standardization on which to build an IS curriculum (i.e., the DPMA guidelines), but do not automatically lead to success. The importance of assessing the needs of the students, in conjunction with the goals of the academic institution, cannot be overstated. The delineation between traditional and non-traditional IS programs does not predicate quality in itself. Indeed, with careful planning and consideration of the critical factors involved, traditional and non-traditional curricula can be equally effective.

#### REFERENCES

- Athey, Susan and John Plotnicki. 1991. A Comparison of Information System Job Requirements in Major Metropolitan Areas. *Interface* 13(4):47-53.
- Bhola, H.S. 1989. World Trends and Issues in Adult Education. London: Kinsley.
- Cassara, Beverly Benner (ed.). 1990. Adult Education in a Multicultural Society. New York: Routledge and Kegan Paul.
- Couger, J. Daniel and Robert A. Zawacki. 1980. Motivating and Managing Computer Personnel. New York: John Wiley and Sons.
- Davis, Gordon B. 1992. Information systems as an academic discipline: explaining the future, Journal of Information Systems Education, 4(4), 2-7.
- Dentzer, Susan. 1993. Bridging the costly skills gap. U.S. News and World Report. 114(3):61-64.
- DPMA. 1991. Information Systems: The DPMA Model Curriculum for a Four Year Undergraduate Degree for the 1990s. Park Ridge, Illinois: Data Processing Management Association.

Greenwald, John. 1992. The great American layoffs. Time. 140(3): 64-65.

Hatch, Richard A., Alexis Koster, and Stanley Marder. 1989. Selection of Programming Languages for the Computer Information Systems Curriculum. Interface: the Computer Education Quarterly 11(2):10-12.

Houle, Cyril O. 1992. The Literature of Adult Education. San Francisco: Jossey-Bass Publishers.

International Labor Office. 1986. Economically Active Population, 1950-2025. Volume IV. Northern America, Europe, Oceania, and USSR. Geneva: International Labor Office.

Jackson, Durward P. 1991. Curriculum Design and the Marketplace for MIS Professionals. Interface 13(4):2-8.

Johnston, William B. 1991. Global Work Force 2000: The New World Labor Market. Harvard Business Review 69(2):115-127.

Katz, Adolph I. and Robert L. DeMichiell. 1992. Information Systems Undergraduate Curricula: Trends for the 1990s. Interface: the Computer Education Quarterly 14(1):2-7.

Larabee J. F. 1992. Incorporating communication skills into a management information systems course, Journal of Information Systems Education, 4(3), 26-31.

Litecky, Chuck and Kirk Arnett. 1992. Job Skill Advertisements and the MIS Curriculum: A Market-Oriented Approach. Interface 14(4):45-52.

Longenecker Jr., Herbert E. and David L. Feinstein. 1991. A Survey of Undergraduate Programs of Information Systems in the U.S. and Canada. Journal of Information Systems Education. 3(1):8-13.

Mackowiak, K. 1991. Skills Required and Jobs Available for CIS Majors. Interface 13(4):9-14.

Markey, J.T. and William Parks. 1989. Occupational change: Pursuing a different kind of work. Monthly Labor Review. 112(9): 3-12.

Nagarajan, N. 1989. MIS majors: Can they meet business expectations? Computerworld, August 7, 21.

Nantz, K. S. 1992. The effectiveness of model curricula in addressing skills needed by information systems students, Interface: The Computer Education Quarterly, 14(2), 2-7.

Pierce, G. E. 1993. Clinton, the high technology president? Electronic News. 39(1946): 8.

Pollack, T. H. 1990. The MIS curriculum: Which competencies are really important to business professionals? Proceedings of the Eighth Information Systems Education Conference, Chicago, Oct. 13-14, 76-80.

Rifkin, G. 1987. MIS courses fall short, Computerworld, June 22, 70-71.

Schrage, J. F. and R. A. Schultheis. 1985. Nontraditional instruction in information systems: Selected concerns and experiences. Interface: The Computer Education Quarterly, Winter, 6(4), 28-33.

Sumner, M., R. Klepper, and R. Schultheis 1990. An assessment of the attitudes of graduates and employers toward competencies need for entry-level MIS positions, Proceedings of the Eighth Information

Systems Education Conference, Chicago, Oct. 13-14, 129-134.

Thomas, Alan and Edward W. Ploman (eds.). 1986. Learning and Development: A Global Perspective. Toronto: Ontario Institute for Studies in Education.

U.S. Department of Labor. 1990. The Fastest Growing Occupations, 1988-2000. The Occupational Outlook Quarterly. Spring. Washington: D.C.: U.S. Department of Labor.

University of Redlands. 1992. "1992 Curriculum Proposal: Report of the IS Program Curriculum Review Committee." Redlands, California: University of Redlands.

Wiersba, R. K. 1992. Business information systems in transition--the challenge for academia, The Journal of Computer Information Systems, 32(2), 50-55.

Yellen, Richard E. and Thomas C. Richards. 1992. The DPMA's 1990 Model Curriculum and Information Systems Programs, A Comparison. International Business Schools Computer Quarterly 4(2):43-47.

**Table 1. Course Sequence for 1993 Undergraduate Information Systems Curriculum, University of Redlands**

Course Sequence		
PREREQUISITE COURSES	Units	Weeks
Introduction to Information Systems	3	6
Introduction to PASCAL Programming Techniques	3	6
<b>TOTAL</b>	<b>6</b>	<b>12</b>
IS Program Organization Meeting (1 week)		1
<b>SEMESTER I (28 weeks, 14 units)</b>		
Philosophical Foundations of Management	6	12
Management Theory and Practice	3	6
COBOL Programming Techniques	2	4
Telecommunications	3	6
<b>SEMESTER II (30 weeks, 15 units)</b>		
Management of Organizational Behavior	3	6
Database Concepts	4	8
Political and Business Economics	3	6
Systems Analysis and Design	3	10
<b>SEMESTER III (20 weeks, 13 units)</b>		
Applied Software Development Project I	3	*
Accounting for Managers	3	6
Business Statistics	4	8
Financial Management	3	6
<b>SEMESTER IV (18 weeks, 12 units)</b>		
Applied Software Development Project II	3	*
Computer Ethics	3	6
Production and Operations Management	3	6
Management and Decision Systems	3	6
<b>TOTAL</b>	<b>54</b>	<b>97</b>

\* Scheduled over entire semester

# Are We Really Educating IS Graduates With the Communications and Problem Solving Skills to the Satisfaction of the IS Industry?

**Panel:** Herbert E. Longenecker, Jr., Panel Chair  
Carl Clavadetscher  
Eli Cohen  
David Feinstein  
Mary Jo Haught  
Mary Sumner  
Robert Zant

## **Background:**

There have been surveys of employers of IS graduates that suggest communications and problem solving are the most essential skills required of IS professionals. The DPMA IS'90 and IS'93 curriculum models spend considerable emphasis in addressing these expectations through the development of formally stated cognitive performance metrics.

It is an explicit suggestion of the two curriculum models that the Bloom applications knowledge recommended for IS graduates can in fact only be demonstrated by explicit domain specific problem solving involving appropriate communication. Furthermore, these advanced levels can only occur through a strategically sequenced spirally developed educational process of lectures and structured laboratory sequences.

## **Questions:**

It is proposed that the panel make presentations of new survey data and conduct discussions to address the following questions. Each panel member will make individual short introductory presentations. Then, an additional round of follow-up discussion by the panel will clarify issues. Audience participation can then expand the concerns of the panel.

1. How important are problem solving and communication skills in the hiring of IS graduates?
2. Is the necessity for communications and problem solving adequately represented and handled in IS'90 and IS'93 recommendations?
3. What is the current status of implementing the recommendations of IS'90 and IS'93 in academia?
4. Do students actually learn the recommended communication and problem solving skills at applications knowledge level?
5. Once students are hired, are problem solving and communications skills really effective?

# A Project-Intensive Introductory Object-Oriented Programming Course

Billy B. L. Lim  
Applied Computer Science Department  
Illinois State University  
Normal, IL 61761.  
BLLIM@ILSTU.BITNET

## Abstract

Object-oriented technology is an emerging technology that has received much attention in both the software industry and the information system/computer science curricula. This paper describes a project-oriented introductory course in object-orientation with emphasis on object-oriented programming. The approaches used and the successful experience in experimenting with various supporting tools are presented.

## 1. Introduction

In recent years, *object-oriented* (OO) technology has become one of the dominant technologies in the computing industry. In a recent survey, it was reported that over 75% of the Fortune 100 companies have adopted OO technology to some degree for their computing needs [Object 92]. To reflect the advances in OO technology and its acceptance by the computing industry [OOPSLA 86-92], object-orientation has been proposed to be integrated into computer science/information systems curricula in the last few years [e.g., Pugh 87, Temte 91, Bellin 92].

While many of the proposals have shared the experiences gained in teaching an OO course and have outlined the "big picture" of the course taught, few have identified the "smaller pieces" and many do not detail how the concepts of OO get reinforced through class activities. This paper describes a *project-oriented* approach to an introductory object-oriented programming (OOP) course and discusses the successful experience in adopting the approach to tackle the aforementioned issues that have not been addressed.

A project-oriented course is different from a "lecture-oriented" one in that in the former, in addition to introducing the fundamental concepts of OOP through traditional lectures, a significant portion of the course is directed

towards the reinforcement of the fundamentals through class activities. These include home exercises that practice "programming in the tiny" (e.g., modify an array class such that it now performs bound checking), programming assignments that can be viewed as a study of "programming in the small" (e.g., develop a simple class library that supports matrix operations), and a course project that involves "programming in the large" (e.g., design and implement a full-fledged database-related application).

In addition to a discussion of the above homework, assignment, and project, collectively referred to as *project* in the term project-oriented approach, this paper also describes the environment employed in implementing the course and some of the implementing issues that have surfaced and been dealt with. Computing resources that are available to educational institutions (at discount) are also identified.

The remainder of this paper is organized as follows. Section 2 gives an overview of the course content. The various homeworks and assignments, the course project, and the experiences gained are described in Section 3. Section 4 discusses the environment used. Finally, Section 5 gives the summary and conclusions.

## 2. Overview of Course Content

The course offered is a semester course that targets the junior and senior students. Students are expected to have as prerequisite a data structure course and the language C is recommended (as it is the "basis" of C++, one of the languages used in the course) but not required.

There are five major components to this course (see Appendix for the course outline). First, the fundamental concepts of object-orientation (i.e., objects, classes, OOP vs. procedural programming, encapsulation, polymorphism, and inheritance) are taught. This component of the course is programming language independent, as the emphasis is on the underlying OO concepts and not specific language implementation. Second, C++ is used as the language for illustrating the concepts discussed in the first part of the course. Here, students are reminded of the ways that things were done in languages like C, Pascal, and PL/I (e.g., *struct* in C compared with *class* in C++). Other C++ specific features (e.g., friend and inline functions) are also covered in this portion of the course. Third, Smalltalk, a "pure" OOP language, is compared and contrasted to C++, a hybrid OOP language, in how they support the object-oriented paradigm. Other than the "pureness" of the language, the other topics discussed include the binding times and the program development environments. Fourth, the design aspects of OO are overviewed. These include an introduction to OO analysis and design using CRC (for Class-Responsibility-Collaboration) cards [Beck 89] and an investigation into class libraries design. Lastly, an overview of other aspects of object-oriented technology such as object-oriented databases and interfaces are presented.

The requirements of the course consist of 2 tests (30%), several homeworks and programming assignments (30%), and a comprehensive group project (40%). As the weights show, the homeworks, programming assignments and the project represent a significant portion of the requirements and thus they, rightly, represent the majority of students' class activities. These requirements are discussed in the next section.

## 3. Homeworks, Assignments and Project

### 3.1 Homeworks and Assignments

As a prelude to the group project, homeworks and programming assignments are given to reinforce the concepts presented in class. They roughly correspond to "programming in the tiny" and "programming in the small," respectively.

In the homeworks, the emphasis is to get students to work on a topic that has just been discussed so that while the materials is still fresh in their minds, the reinforcement exercise can be completed quickly (i.e., in days). For example, immediately following the discussion of operator overloading, a form of polymorphism, students are asked to complete a homework exercise that involves the modification of an array class to include bound checking.

In the programming assignments, students are typically required to first complete a series of "operations" using the traditional (i.e., *procedural*) paradigm. Then in subsequent exercises, they are required to re-implement the same set of operations using the *object-oriented* paradigm. This permits clear contrast of the two paradigms and many students have expressed their appreciation as a result this comparison.

An example of the types of operations that had been used in this course is the support for complex matrix operations (e.g., assignment, I/O, multiplication, addition, etc.). This exercise allows the three defining characteristics of OO (i.e., encapsulation, polymorphism, and inheritance) to be easily exhibited. The concept of encapsulation can be reinforced by making the complex matrix an abstract data type (ADT) through the use of the *class* feature in C++. With that, both the data/state and procedures/behaviors are encapsulated and only a predefined set of operations can be carried out on an object that belongs to the class.

Polymorphism is easily reinforceable when the operations are implemented. Through operator overloading, operator like "+" can be used for complex matrix addition, complex number addition, and also integer/real addition. The actual operation carried out is, of course, determined by the type of the operand/object involved.

As for inheritance, an "artificial" superclass with minimal functionality (e.g., supports only the data structure and not the operations) can be created so that a real subclass that is to have all the behaviors can inherit from this artificial superclass. This exercise, while artificial, gives the students an opportunity to have an hands-on experience in dealing with class hierarchy and inheritance feature.

### 3.2 Course Project

The course project gives the students an opportunity to practice OO concepts by *designing and implementing either a class library or an application that makes use of class libraries* for a subject area of interest (e.g., a game, a CS/IS course, a business or organization, etc.).

The central theme of the project is *reusability*. If a student is designing a class library, he/she should bear in mind that what he/she implements is to be used by the library *users* and thus they should only see the interface portions of the classes that have been developed. Implementation details should be hidden away, i.e., *encapsulation* of both the data and procedures should be provided. The student should also practice the concepts of *polymorphism* and *inheritance* since the class library may also be reused by other library designers for refining the capabilities that have been provided. These make the class library easier to use and make the interface more consistent. The emphasis of this type of project is on the functionality, ease of use, as well as the design of the library. This is where the design aspects of the course are brought into the classroom for discussion.

On the other hand, if a student is using class libraries to develop some application of interest, then the emphasis of the project is on the overall functionality of the application as well as the user interface of the application. The user interface should be graphical in that it should support pull-down menus, pop-up windows, dialog boxes, mouse, etc., all of which are available through class libraries that are provided to the students.

The aforementioned project spans most of the parts of a software development life cycle (SDLC) - from problem definition through specification of requirements to the design and (partial) implementation of the application system. The project is carried out over the

course of the semester in milestones, at which the cumulative project documentation is turned in, graded and returned.

So far, more than twenty five projects have gone through the course, each with one to three members. They represent a wide variety of topics, ranging from the development of pedagogical tools to personal interest related applications. Some of which include a pen-based Solitaire game, multimedia tool for teaching OO concepts [Chang 93], graphical tools for teaching data structures [Lim 93] and database design [Lim 92].

### 3.3 Classroom Experience

Among the lessons learned is that students need as many "programming in the tiny" exercises as possible due to the steep learning curve for the understanding of OO paradigm. Thus, before they can put together a large piece of software (i.e., the project), they need to have the confidence in tackling smaller ones first.

Also, with respect to implementing the project, some of the problems encountered include the lack of time in completing the prototype and the tremendous amount of efforts needed and difficulties faced in mastering the use of the class libraries supplied. Future considerations include putting less emphasis on the earlier phases of the project (to solve the former problem) and orienting the students on the class libraries to be used in the development (to solve the latter).

## 4. Environment

The computing environment used in this course is PC-based. (While Eiffel running on the Sun's is available and experimentation of its unique support for *assertions* is encouraged, no student chose the platform in their projects.) For the C++ component of the course, Turbo/Borland C++ from Borland International was used for the assignments. Many project groups have also chosen this, together with Turbo Vision/ObjectWindows, as their implementation platform. For the Smalltalk component, Digitalk's Smalltalk for Windows was used for illustration. It was also used in some of the group projects. Lastly, Zinc Interface Library has also been used by some of the groups in place of Turbo Vision for building their user interface.



All of the above software packages are available at educational discount to educational institutions. The ones from Borland International are especially great buys. The C++ package also includes video tapes that can be used to supplement class notes. One language/environment that will be considered in future offering of the course is a language called Object-Oriented Turing [Mancoridis 93a, PL Panel 93, Mancoridis 93b] for its comprehensive treatment of software development process.

## 5. Summary and Conclusions

This paper describes an undergraduate introductory object-oriented course that has been offered to juniors and seniors. It stresses the programming assignment and project activities of the course and it discusses the insights of teaching the course.

From student feedback (through student evaluations) and classroom experience, it is felt that a course organized in this manner permits maximum transfer of the underlying concepts behind object-orientation. An equally important lesson that is learned is the retention of the concepts taught through the reinforcement in the programming assignments and the comprehensive course project.

It is felt that a project-oriented approach to teaching OOP is effective and the author strongly recommends that it be considered as an approach to integrating OO concepts into a computer science/information system curriculum.

## References

- [Bellin 92] Bellin, David., "A Seminar Course in Object-Oriented Programming," *23rd SIGCSE Technical Symposium on Computer Science Education*, Kansas City, MI, March, 1992.
- [Beck 89] Beck, K., Cunningham, W., "A Laboratory for Teaching Object-Oriented Thinking," *ACM International Conference on Object-Oriented Programming: Systems, Languages, and Applications*, October, 1989.
- [Chang 93] Chang, C., C., Rendon, G., Lim, Billy B. L., "TOOP: A Multimedia Tutorial for Object-Oriented Programming," *1993 Midwest Computer Conference*, Whitewater, WI, March, 1993.
- [Lim 92] Lim, Billy B. L., Hunter, R., "DBTool: A Graphical Database Design Tool for an Undergraduate Database Course," *ACM SIGCSE Bulletin*, 23:1, March, 1992.
- [Lim 93] Lim, Billy B. L., Wang, J., Kohlbrecker, C., Jason, J., "IDST: A GUI-Based Tutoring System for Learning Data Structures," *ISECON '93*, Phoenix, Arizona, November, 1993.
- [Mancoridis 93a] Mancoridis, S., Holt, R., Penney, D., "A 'Curriculum Cycle' Environment for Teaching Programming," *24th SIGCSE Technical Symposium on Computer Science Education*, Indianapolis, IN, February, 1993.
- [Mancoridis 93b] Mancoridis, S., Holt, R., Penney, D., "A Conceptual Framework for Software Development," *21st Annual Computer Science Conference*, Indianapolis, IN, February, 1993.
- [Object 92] Executive Summary, *Object Magazine*, July-August, Vol. 2, No. 2, 1992.
- [OOPSLA 86-92] Annual ACM International Conference on Object-Oriented Programming: Systems, Languages, and Applications, 1986-1992.
- [PL Panel 93] Panel Session: Issues in the Choice of Programming Language for CS 1, *24th SIGCSE Technical Symposium on Computer Science Education*, Indianapolis, IN, February, 1993.

- [Pugh 87] Pugh, J. R., LaLonde, W. R., Thomas, D. A., "Introducing Object-Oriented Programming into Computer Science Curriculum," *18th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, MI, February, 1987.
- [Temte 91] Temte, M. C., "Let's Begin Introducing the Object-Oriented Paradigms," *22nd SIGCSE Technical Symposium on Computer Science Education*, San Antonio, TX, March, 1992.

- Introduction to O-O Analysis & Design, CRC Cards
- Class Library Design
- Reusability Design
- ◆ Other Object-Oriented Topics (if schedule permits)
  - OO Implementation, OO Languages
  - Object-Oriented Databases (OODBMS)
  - Object-Oriented Graphical User Interfaces

## Appendix

### Course Content:

- ◆ Introduction to the Fundamentals of Object-Oriented
  - Objects
  - Classes
  - OOP vs. Procedural Programming
  - Data Abstraction and Encapsulation
  - Polymorphism and Late Binding
  - Inheritance
- ◆ C++
  - Introduction (C++ as a Better C, History, Design Goals, etc.)
  - C++ Classes & Objects
  - Member Functions, Virtual Functions, & Friend Functions
  - Function & Operator Overloading
  - Constructors & Destructors
  - Derived Classes
  - C++ Stream I/O and Library
- ◆ Smalltalk
  - Introduction to Smalltalk Programming Environment
  - Objects, Variables and Assignment, Control Structures
  - Message Types, Message-Passing
  - Self vs. Super
  - Browser/Inspector/Debugger
  - Smalltalk Class Library
- ◆ Design

# Integrating Object-Oriented Technology into a Database Course using SOD - a Simple Object Database

Conrad T. K. Chang  
Department of Computer Science and Engineering  
University of Illinois at Urbana-Champaign  
Urbana, IL 61825  
chang@osiris.cso.uiuc.edu

Billy B. L. Lim  
Applied Computer Science Department  
Illinois State University  
Normal, IL 61761.  
BLLIM@ILSTU.BITNET

## Abstract

As object-oriented (OO) technology becomes more and more popular, the attention given to object-oriented database management systems (OODBMSs) has also risen. While the database industry has been preparing itself for this transition, the information system curricula have not been geared up to deal with this change. This paper describes a simple OODBMS that has been developed to serve as a tool to help integrate OO technology into database courses in information system curricula.

## 1. Introduction

In recent years, *object-oriented* (OO) technology has become one of the dominant technologies in the computing industry. In a recent survey, it was reported that over 75% of the Fortune 100 companies have adopted OO technology to some degree for their computing needs [Object 92]. To reflect the advances in OO technology and its acceptance by the computing industry [OOP 86-92], object-orientation has been integrated into computer science/information systems curricula in the last few years [e.g., Pugh 87, Temte 91, Lim 93]. While OO has been steadily incorporated into the curricula, the incorporation has not been with database courses. This paper describes an OODBMS called SOD (Simple Object Database) that has been developed to aid the integration process.

SOD is an experimental OODBMS designed and developed to serve two purposes. First, the project provides great opportunity to gain a deeper understanding of OODBMS. It allows one to know more about potential implementation difficulties of an actual OODBMS and to be more aware of the potential applications of such a system. Second, it is hoped that others can use this system as an

educational tool. It is the latter that this paper is addressing.

First generation commercial OODBMSs are often very expensive. For most educational environments, it is not feasible to purchase these products for large-scale teaching or experimental purpose. Thus, most students are learning about OODBMS only through textbook readings. However, to teach the concepts effectively, a combination of textbook readings and experiences with an OODBMS is a must. SOD is intended to be used as a teaching tool to fill this gap. Typical users include students and teachers in an educational environment such as a university. Students can use this system for experimental purposes while teachers can use this system as a demonstration of OODBMS concepts. SOD is currently being experimented in the author's institution.

The remainder of this paper is organized as follows. Section 2 gives an overview of SOD. The functionality and user interface of SOD are discussed in this section. The inner workings of SOD is detailed in Section 3. Section 4 discusses the environment used. Finally, Section 5 gives the summary and conclusions.

## 2. Overview of SOD

### 2.1 Background

While there is lack of standards with regard to OODBMS concepts (with the exception of the effort by Atkinson et. al. [Atkin 91]), it is commonly agreed that OO databases should have the characteristics of both object-orientation and databases. Object-orientation includes abstract data types (classes), complex objects, inheritance, polymorphism, and object identity. Database capabilities include the traditional issues like persistence, transaction management, concurrency control, recovery, querying, integrity, and security.

A *class* is an abstract representation of an entity that is of interest to users. An entity can be a car, a person, a student, etc. *Objects* are instances of a class. They are where information is stored. Attributes of a class are like data fields in a record. Information about an entity is stored in these attributes. For example, to store information of a person named Tom, a Person class with attributes Age, Address, Phone number, etc. can be created first. Then an object can be created from this class for storing Tom's information.

*Inheritance* is a powerful technique for organizing complex class structures. It allows the construction of new classes on top of existing class hierarchies. Through inheritance richer semantic relationships among entities in an object space can be expressed directly and naturally. In SOD, a class definition can be inherited from its superclass. For instance, if a Person class contains the attribute Name, then a Student class with attribute GPA can be defined as a subclass of the Person class. As a result, the complete definition of the Student class will include all attributes inherited from the Person class, in addition to its own attributes (see figure 1).

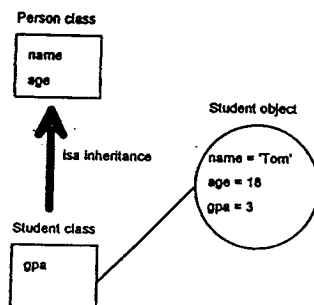


Figure 1: Student class inherits from Person class

### 2.2 Functionality

SOD allows users to define, manipulate, and query data in an object-oriented fashion. Relationships can be defined to link objects. When linked, the object containing the link can be considered a *complex object*. All the relationships and data are kept in a persistent data store for processing. SOD is also designed to be easy to use with the implementation of mouse support, on-line documentation, and pull down menus.

SOD has the following features. It supports

- class definition
- single class inheritance
- multivalued attributes
- object identity
- complex objects
- behaviors
- on-line help
- querying of database by user-specified conditions
- easy to use menu-driven graphical user interface (GUI)

### 2.3 Interface

SOD has a character-based GUI. Overall, the user interface is designed in such a way that users are guided through each operation (e.g., create a class, issue a query). Windows are used to direct user's attention to a particular part of the screen. Extensive dialog boxes and other graphical elements (e.g., check box, push button, scroll list) are used for interacting with users to create a more intuitive environment. Furthermore, the system provides all the valid options at any given point during an operation. Invalid options are always disabled and cannot be selected. If a mistake is made, users always can cancel their actions. A Cancel button is present in all dialog boxes.

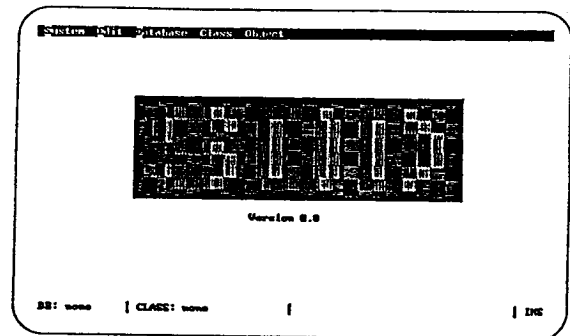


Figure 2: SOD system screen

In SOD, the screen is divided into three parts: the system menu bar at the top, the status bar at the bottom, and the work area in the middle. The system menu bar is where users select functions to

perform. The status bar displays various information about the current database. The work area can be used for many purposes (see figure 2).

### System Menu Bar

The system menu bar is a pull-down menu bar with five options: System, Edit, Database, Class, and Object. In general, the System menu contains options for displaying information about the system and for changing the environment setting such as clock and display mode. The Edit option contains the standard editing features like cut, paste, find, etc. The Database menu contains options for creating, opening, closing, packing and deleting a database. Users can use the Class menu to perform operations on classes. Finally, the Object menu allows users to locate an object or run the behavior of an object.

The status bar displays the following information: current database name, current class name, number of objects in the current class, and the insert mode (on or off). Information on the status bar will be changed according to the current system status. If a user changes the current class, then the current class name and the number of object in the class will also be changed on the status bar.

The work area is used for many purposes by the system. Section 3 shows much of the uses. The system also provides an on-line help documentation. At any point, users can hit F1 to see a list of help topics. The help topics include a list of error messages, and descriptions of each option on the system menu bar.

Each of the five options from the system menu bar and their corresponding sub-options are shown in Figures 3 through 7. The System option contains system related settings, not discussed here. The Edit option is self-explanatory, and the rest of the options are detailed in Section 3.

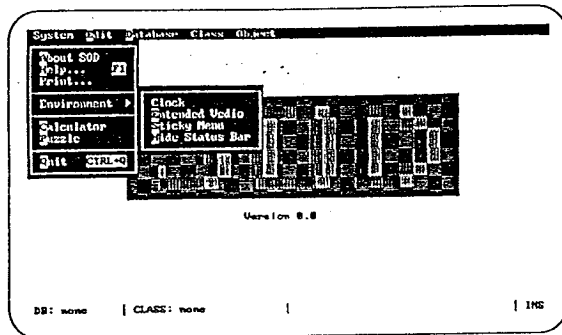


Figure 3: System menu popup

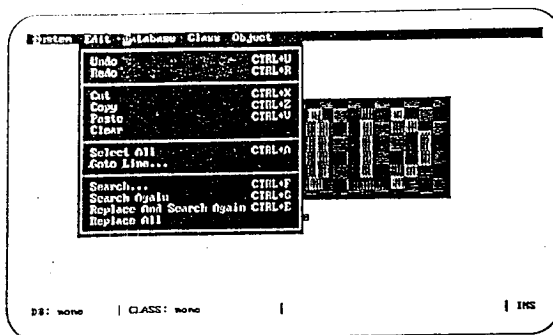


Figure 4: Edit menu popup

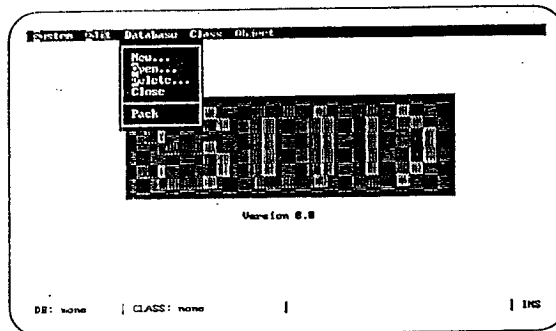


Figure 5: Database menu popup

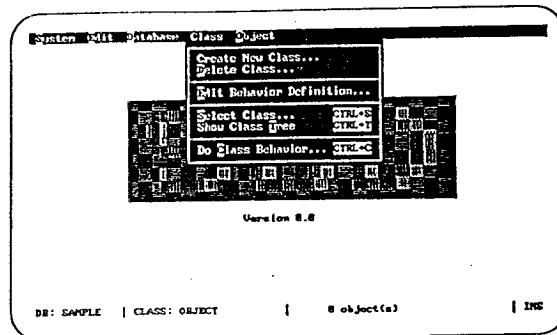


Figure 6: Class menu popup

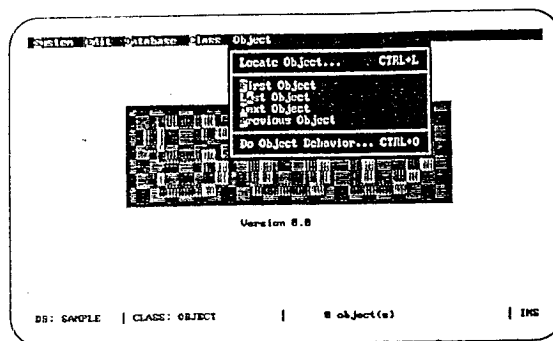


Figure 7: Object menu popup

### 3. Working with SOD

#### 3.1 Working with Databases

In a relational DBMS, a database is a collection of tables. In SOD, it is a collection of classes, objects, and behaviors. When a database is opened, all of its data (i.e., classes, objects, and behaviors) can be accessed. SOD allows multiple databases to exist.

To create a new database, one can choose New from the Database menu. The New Database dialog appears. The default database name is set to NEWDB. To open an existing database, one can choose Open from the Database menu. The Open Database dialog appears. To close a database, one can choose Close from the Database menu. After a database is closed, all classes, objects, and behaviors are also closed and cleared from the system. Therefore, they are no longer accessible. To delete a database, one can choose Delete from the Database menu. The Delete Database dialog appears. Deleting a database deletes all its classes, objects, and behaviors. Once a database is deleted, everything in the database is destroyed.

To pack a database, one can choose Pack from the Database menu. When a user runs the delete object behavior (discussed in Section 3.3) to delete an object, it is not physically removed from the database but simply marked as deleted. To reclaim the space it occupied, the user needs to pack the database. This action removes all deleted objects of any class from the database.

#### 3.2 Working with Classes

Classes define the structure and behavior of objects. They are the blue prints of objects. When a class is defined, attributes are defined for capturing information needed for the entity that the class is representing. These attributes can either be one of the four primitive data types or any class types previously defined. The four primitive data types supported are integer, real, character string, and date.

In addition, an attribute can either contain multiple values or a single value. For example, a phone number attribute of a person object can contain multiple numbers that are of integer type; a course attribute of a student object can contain multiple course objects, which in turns contain course information (see figure 8).

A class can also contain behaviors. These behaviors are the know-how of a class. Everything that one wants to do with a class is done through running these behaviors. For example, to create an

object, one needs to run the Create Object behavior (because the class is the blue print!). This behavior displays the data entry screen and allows users to add attributes' values. Class behaviors can be defined or modified by users at any time to tailor to their needs. Three behaviors are provided for each class by the system. A more detail discussion of behaviors can be found in Section 3.4.

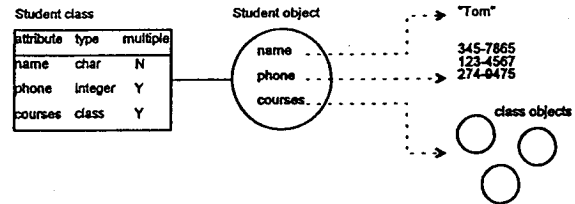


Figure 8: Multivalued attribute of a class

Figures 9 show the dialog box for defining a class.

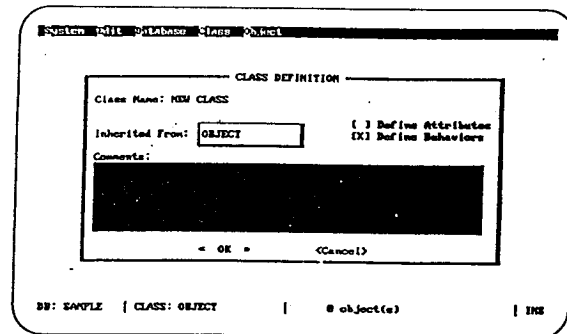


Figure 9: Creating a new class

To define an attribute of a class, one can choose one of the four primitives or a class type. A class type is a class that has been defined. An attribute of a class type can store objects of that class. For example, assume that a Person class has been defined and a Department class is to be created. Now, an attribute of the department class can be defined as a Person class type. In this case, the attribute can be used to store objects (e.g., all employees) from the Person class.

Moreover, an attribute can also be a multivalued attribute. Usually, each attribute can only store one item of data. For example, a Name attribute of string type can only store one name. However, if the attribute is made multivalued, then multiple names can be stored in this attribute. A class type is always a multivalued attribute.

To delete a class, the Class menu provides a Delete Class option. A dialog pops up and asks for

confirmation since deleting a class will delete all of its objects and subclasses! Therefore, it is absolutely important that the users know the consequences.

To change a class, one may choose the Select Class option from the Class menu. The Select Class dialog appears. A list of classes is displayed inside the window (see figure 10).

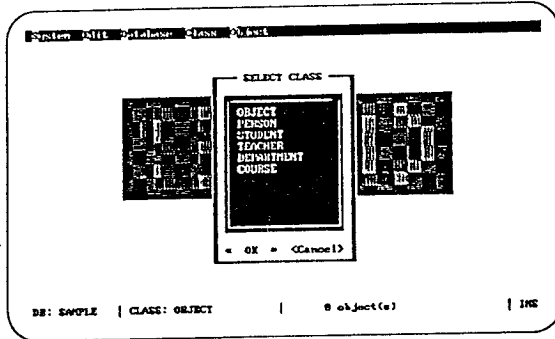


Figure 10: Select a class

To see the inheritance hierarchy, one can choose Show Class Tree from the Class menu. The Class Inheritance Tree window appears. The inheritance tree diagram is displayed inside the window (see figure 11).

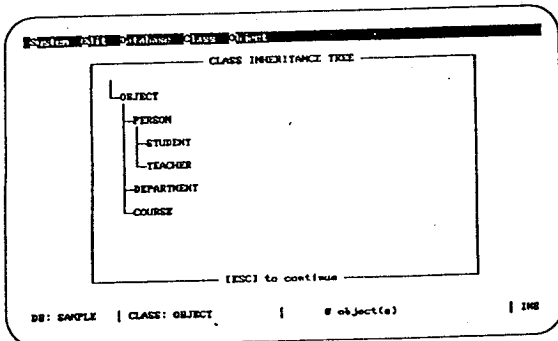


Figure 11: Displaying class inheritance tree

To run a class behavior, one can choose Do Class Behavior from the Class menu. The Do Class Behavior dialog appears (see figure 12). The appropriate class behavior can be chosen from the Behavior scroll list.

### 3.3 Working with Objects

Every object belongs to a class. To work with an object in a particular class, it is necessary that the class is currently selected. This can be done by selecting the class from the Class menu. Once a class is selected, the first object of the class is current. Because there is no particular order in which objects of a class are arranged, the users

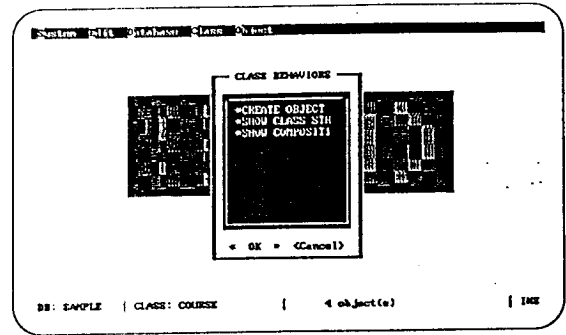


Figure 12: Running a class behavior

should not assume that the first object is always the same object. The order of objects is changed every time an object is added to or deleted from a class. The addition or deletion of objects is done through object behaviors.

In SOD, each object contains not only data, but also the behaviors associated with it. These behaviors are the capabilities of an object - the only things it knows how to do. For example, to modify an object, a user needs to run the Modify Object behavior. This behavior displays the data entry screen and allows the user to change attributes' values. Users can define their own behaviors to tailor to their needs. Two common behaviors are provided for the objects by the system. A more detail discussion of behaviors can be found Section 3.4.

To help navigate among the objects, four options are provided for moving around the objects of a class. These options are First Object, Next Object, Previous Object, and Last Object for going to the first, next, previous, and last object in the current class, respectively.

Besides going through objects in a class sequentially, users can also locate an object or a set of objects with search criteria. After the search conditions are specified, the system searches through all objects of the current class to find a match. If more than one object satisfy the conditions, a list of objects will be displayed in a scroll list. They can then be chosen to be the current object. The process of locating objects is discussed in Section 3.5.

### 3.4 Working with Behaviors

Behaviors are programs associated with each class or object. They represent the *only* protocol that objects and classes understand. There are two types of behavior: Class Behavior and Object Behavior. Class behaviors consist of those behaviors that have effects on classes. For example, showing class structure and creating an object of a class are both

class behaviors. On the other hand, object behaviors act on individual objects. For example, deleting an object is an object behavior. Both behaviors can be defined and modified by users.

SOD provides three class behaviors for each class: Create Object, Show Class Structure, and Show Composition Tree. Two object behaviors are also provided for each object: Modify Object and Delete Object. Users can define their own behaviors (e.g., display all IS majors) by writing scripts in FoxPro, the environment in which SOD is constructed.

Figures 13 through 15 illustrate the use of some of the predefined behaviors.

ATTRIBUTE	TYPE	SIZE	MULTIPLE	CLASS	FROM CLASS
COURSE NUMBER	INTEGER	5	N	N	COURSE
DAYS OF WEEK	STRING	5	N	N	COURSE
START TIME	STRING	4	N	N	COURSE
END TIME	STRING	4	N	N	COURSE
STUDENTS	STUDENT	6	Y	Y	COURSE
TEACHER	TEACHER	6	Y	Y	COURSE
DEPARTMENT	DEPARTMENT	6	Y	Y	COURSE

Figure 13: Displaying class structure

```

Class = COURSE * contains
  Class = STUDENT *
  Class = TEACHER *
  Class = DEPARTMENT * contains
    Class = TEACHER *
  
```

Figure 14: Displaying class composition diagram

Figure 15: Defining behaviors

### 3.5 Finding Objects

SOD can search for objects in a class based on the criteria given by users. With the criteria, the system

searches through all objects of the current class to find a match. Criteria may contain multiple conditions, each of which consists of

Logical Operator + Attribute Name + Relational Operator + Value

where

Logical operators	AND, OR
Relational operators	=, != (not equal to), <, >, <=, >=, ~ (approximately equal to)
Attribute name	attribute name of an object used for matching
Value	user specified value used for matching

The process of specifying conditions is done through a dialog. Users are presented with the list of logical operators, attribute names, and relational operators. They can then enter values for the attributes to match. The list of attribute names is generated each time for an object. It contains non-class type attributes from all classes contained by the current class. For example, if you want to search objects in an X class that contains a Y class, which in turn contains a Z class, SOD would collect all non-class type attributes from all three classes into a single list. A qualifier is added to the attribute name to indicate its origin if this attribute does not belong directly to the current class (see figure 16).

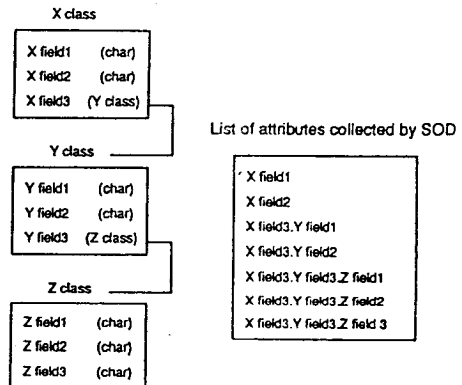


Figure 16: Attribute collected from a class

In addition, SOD allows users to save their conditions to a query. Later, users can restore this query for locating objects. (The Save, Restore, and Erase query options are presented in the Expression dialog, given below.) To locate an object, attributes of the object to display when a match is found need to be selected. Then, the criteria that the searching is based on has to be specified. After that, the search can begin. If more than one object satisfy the criteria, a list of these objects is displayed in a scroll list for user to choose from. Figures 17 through 19 illustrate the process.



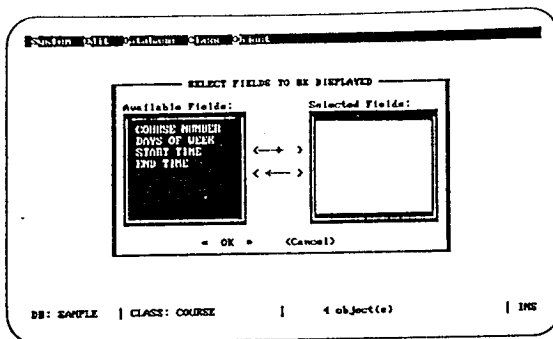


Figure 17: Selecting fields to display

After you select the fields to display, a dialog pops up for you to specify your criteria.

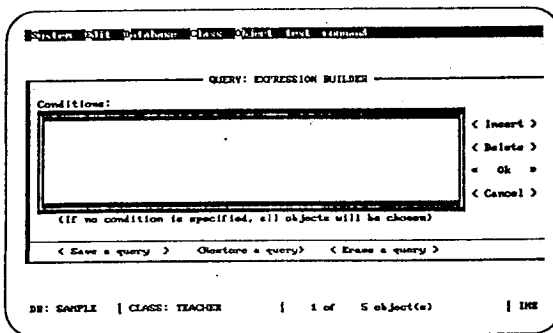


Figure 18: Defining conditions for a search

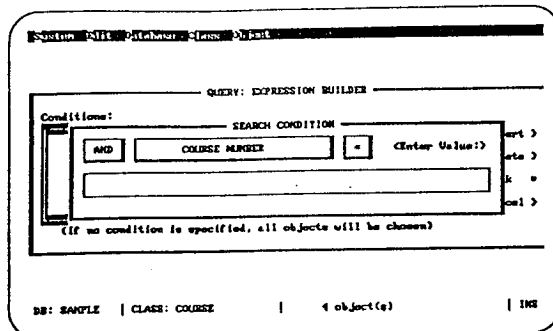


Figure 19: Specifying components of a condition

#### 4. Environment

The SOD system is developed using the FoxPro language in the FoxPro 2.0 environment. The entire system consists of approximately eight thousands lines of code. At least 640K of memory is needed to run SOD because of its extensive use of recursive functions.

The FoxPro language is a dBASE type language supported by the FoxPro relational DBMS. Among the features provided are the support for recursive functions, dynamic binding of variable at run-time, dynamic memory management, and primitive file I/O operations. All of these features are heavily

utilized in the development of SOD and contribute to the success of the project.

#### 5. Summary and Conclusions

This paper describes a simple OODBMS called SOD that can be used to integrate OO technology into database courses in an information system curriculum.

It is felt that the development of SOD is worthwhile and valuable. The many distinctive features of OODBMS that SOD provides enable one to teach and learn the fundamental concepts behind OODBMS effectively and economically.

SOD provides an easy-to-use, learn-at-your-own-pace application that students may use to develop their own database applications. Since SOD is "home-grown," it may be freely distributed to students for use outside of the lab.

It is felt that a tool like SOD is needed to effectively teach OODBMS and the author strongly recommend that it be considered as an aid to integrating OO concepts into a database course.

#### References

- [Atkin 91] Atkinson et. al. "Object-Oriented Database Systems Manifesto," *Proc. of the 1st International Conference on Deductive and Object-Oriented Databases*, North-Holland, 1991.
- [Lim 93] Lim, Billy B. L., "A Project-Intensive Introductory Object-Oriented Programming Course," submitted to *ISECON '93*, Phoenix, Arizona, November, 1993.
- [Object 92] Executive Summary, *Object Magazine*, July-August, Vol. 2, No. 2, 1992.
- [OOP 86-92] Annual ACM International Conference on Object-Oriented Programming: Systems, Languages, and Applications, 1986-1992.
- [Pugh 87] Pugh, J. R., LaLonde, W. R., Thomas, D. A., "Introducing Object-Oriented Programming into Computer Science Curriculum," *18th SIGCSE Technical Symposium on Computer Science Education*, St. Louis, MI, February, 1987.
- [Temte 91] Temte, M. C., "Let's Begin Introducing the Object-Oriented Paradigms," *22nd SIGCSE*, San Antonio, TX, March, 1992.

# Integrating Object Orientation in the IS Curriculum

Les Waguespack, Ph.D.

Computer Information Systems Department, Bentley College

175 Forest Street, Waltham, MA 02154-4705

(617) 891-2908

EMAIL: LWAGUESPACK@BENTLEY.edu

## Abstract

Object orientation has evolved beyond the laboratory and into production information system applications. It is particularly intriguing to IS managers because of its potential for enhancing software reuse. Object orientation has not yet found a home in information system curricula which traditionally focus less on the underlying science and more on the practical use of technology in application development. This paper addresses the questions of why, where and how object orientation may be integrated into the IS curriculum and what tradeoffs must be contemplated in the choices. It describes three approaches.

## 0.0 Introduction

Entering the 1990's there is no less concern over software productivity and reliability than when the software crisis was "discovered" in the early 1970's. In a fashion akin to the emergence of the relational data model as a driving force in the evolution of database technology, object orientation promises to drive the evolution of software development by offering new paradigms for analysis, design, programming and data management.

The underlying concepts of object orientation were born in the simulation programming language SIMULA 67 [Dahl 66], explored and enriched in Smalltalk [Goldberg 83], and are now commercialized and mass distributed in C++ [Stroustrup 86] and object dialects of Pascal [Tesler 85, Symantec 91, Borland 92].<sup>1</sup> They portend significant changes in the system developer's mind set of a generation of IS analysts, designers and programmers who have been raised on the "structured-approach" to systems development. Early experience with object orientation in the IS world seems to suggest that object orientation is orthogonal to structured systems modeling. Therefore, it is unlikely that object orientation will diffuse in the industry without some formal preparation and education.

Object orientation material is creeping into the CIS curriculum. In our program there are learning units on object orientation in programming, data structures, data management, systems analysis and design and software engineering. The treatment of object orientation tends to be low level and introductory in nature and in many cases is repetitious from course to course. To better serve the needs of the students, we need a plan for integrating object orientation.

This paper explores alternative approaches for preparing and educating new practitioners, the undergraduate and graduate students of IS in the 90's. It assumes that the reader is familiar with the concepts of object orientation and how the object orientation paradigm differs from the structured techniques. The first section explores the character of the object oriented system model and the degree to which it is comparable to existing system models in IS. Section 2 examines object orientation in cross sections consistent with the decomposition of software development normally found in IS curricula. Section 3 presents three alternative strategies for integrating object orientation into an IS curriculum offering benefits, costs, and risks. The last section summarizes the issues facing us as IS educators.

---

<sup>1</sup> At the time of this writing there have been rumors of Object-COBOL.

## 1.0 System Models in Development and Curriculum

The need for integrating object orientation in the IS curriculum hinges on the question: "What characterization of object orientation is of primary value to IS professionals?" Just as an algorithmic programming view underlies the CS curriculum, the systems analysis and modeling view drives IS practice and curriculum. We believe the answer lies in the system model that object oriented technology most closely supports.<sup>2</sup>

To a large extent the system model that underlies (or at least predominates) the practice tips the scales in technology and academic selection and sequencing decisions. The system modeling approaches may be abstracted into three categories: process centered, data centered and object centered. An examination of the degree of compatibility or interchangeability there may be between the system modeling approaches provides a framework for understanding why these models impact curricula and how object orientation can be integrated.

### 1.1 System Models

The structured technologies favor a process oriented view of the world in which abstraction is rooted in the aggregation of sub-processes that give definition to larger abstract processes. Top down structured decomposition is a "divide and conquer" methodology devised to identify the boundaries between sub-processes describing the problem as a collection of composite sub-processes. The architectural element of structured technologies is the process, therefore structured approaches are *process driven*.

**Process driven modeling :** that disposition in thinking that focuses initially and primarily upon the actions observed in a situation within the environment and then proceeds to determine the data which must flow through and among these actions to completely model the situation.

By contrast, entity relationship modeling of

<sup>2</sup>This is in some contrast to the role that object orientation plays in computer science curricula. Computer science programs have tended to focus on object orientation as a programming mechanism rather than as a system model.

systems focuses on the entities about which the enterprise needs to store data. The structure of the system depends on the interrelationships of these data in terms of the entities, their associations, and referential relationships.<sup>3</sup> The actions which occur in this model serve only to transform the data from one (stable) state to another. This approach is *data driven*.

**Data driven modeling :** that disposition in thinking that focuses initially and primarily upon the data components of a situation within the environment and then proceeds to define all actions in that environment that set and change those data components.

Object orientation combines aspects of both the process driven and data driven by balancing the characterization of the system model with state information (instance variables) and particularized state transformations (methods of that object). The resulting characterization focuses on how the object behaves when stimulated with certain messages. A system is modeled as a collection of interacting objects communicating via messages. This approach may be called *behavior driven*.

**Behavior driven :** that disposition in thinking that attempts to perceive patterns of behavior in a situation within an environment along with the stimuli that evoke those behaviors and then proceeds to discover the state data and actions that must be present to sustain those patterns of behavior.

Since these three modeling approaches to the problem domain focus on system issues from three different perspectives, they lead to distinct approaches to system documentation, diagrammatic depiction, methods of domain analysis, and styles of software architecture. Efforts to interchange domain model specifications based on the different

<sup>3</sup>Entity-Relationship modeling is representative of a variety of data driven system models. Information Modeling is another example in this group.

approaches have met with limited success [Alabiso 88, Ward 89, Fichman 92, Korson 92]. It has proven particularly difficult to map implementation stage domain information in one characterization to the analysis or design information captured in another. The predominant approach in systems development has been to choose one system modeling characterization and maintain that choice throughout development.

## 1.2 Model Influence on Curriculum

Just as it is difficult for a development organization to support multiple development paradigms based on different underlying system models, it is difficult to fully support multiple system models in an IS curriculum.<sup>4</sup> Because IS education is closely linked with professional practice issues (problems and needs as opposed to theory and science as is computer science), it cannot ignore a significant issue of concern to the IS industry. Process driven models led to emphasis of third generation programming languages<sup>5</sup>, structured analysis and structured design in IS curricula. Data driven models complemented the IS curriculum supporting interest in end-user computing, 4GL development tools and relational database.

IS curricula have been influenced by the process and data models. Object orientation represents a new system model for which we must decide how to integrate a behavior driven view of the world into the IS curriculum. Since object orientation is not a variation on process driven or data driven models and the methodologies and tool set that accompany object orientation have very little (if any) overlap with

<sup>4</sup> Computer science curricula seem to be immune to this issue primarily because systems analysis and design are not usually considered core academic areas in the discipline. System models are often introduced as adjuncts to data management or non-procedural programming rather than motivating system perspectives that guide programming decisions. For instance, data abstraction and modularization are usually treated as programming principles rather than system modeling characteristics.

<sup>5</sup> It may be strongly argued that procedural programming languages gave birth to the process driven system model as a natural abstraction of the only known programming paradigm of its time. This situation is also found in data management when the hierarchical and network data models were "invented" to characterize database systems that existed prior to CODASYL.

the other models, object orientation can not be integrated without making room for it in the curriculum. That room comes by tradeoffs: breadth and depth of other topics covered in lectures, budget and support for software tools, and political deference to the system model orientation of the surrounding IS market place.

In order to make room, we must have an idea of the shape and size of the object oriented concepts from which we may choose. This will allow us to measure the cost of what is displaced against the benefit of what is gained. In the following section we present a brief overview of object orientation from an academic sub-discipline perspective.

## 2.0 Facets of Object Orientation

Object orientation has at least four facets: object oriented analysis, object oriented design, object oriented programming, and object oriented database. This section briefly explores the character of each of these facets.

### 2.1 Object Oriented Programming

In many ways object oriented programming represents only another set of language features added to the old standards that have been central to the teaching of programming in the past. Inheritance and polymorphism are added to encapsulation, modularization, and abstraction that characterized the evolution of high-level block structured languages of the 70's. The richness of abstraction that is possible with classes and objects seems to demand a longer (if not steeper) learning curve for student programmers.

There are several wholly object oriented programming languages (Smalltalk [Goldberg 83], Flavors [Moon 86] and Actor [Duff 86] are a few.) But, these have only enjoyed reference as obscure experimental programming languages in the IS domain. In the past decade Pascal has often represented the language of choice for teaching programming because its roots as a pedagogical tool tend to encapsulate the principles of programming eschewing the more arcane language features that characterized commercially oriented dialects. During that same decade COBOL's status of lingua franca for business oriented application development has been significantly eroded by 4GL's, application generators, microcomputer languages (Lotus 1-2-3, Excel, DBase, Paradox, etc.), and C which is the de facto replacement for assembler language. The object

oriented variations of Pascal and C are not simple extensions of the base languages. C++ in particular, has been described as a completely different language with C as a subset [Ladd 90].

In addition to the added syntax to support objects, the object oriented languages also include "bootstrap" class libraries that provide a foundation of class definitions upon which the programmer "grows" a new application. The class library (which is distinct for each vendor's language product) represents a significant learning investment on its own. Because of these impediments to adoption, most languages are wrapped in a powerful (and usually complex) development environment which supports class library browsing, separate compilation, interactive symbolic debugging, and varying degrees of software version control. In the past most of these "wrapping" features could be dealt with as topics in advanced development or software engineering courses, but in object oriented programming environments these features are necessary "every day" tools.

## 2.2 Object Oriented Analysis

Analysis is the process of formulating a model which captures a problem's characteristics, characteristics which we are interested in studying or controlling. The model consists of parts and their relationships. In this, structured analysis, data modeling and object oriented analysis are the same. However, they model using different modeling elements and a different modeling architecture.

Object oriented analysis characterizes the problem domain in terms of objects and classes rather than processes or data entities. Hence, the object oriented analysis process is different. It asks different questions, for example, "How are these objects alike?" and "How are they different?" The answers to these questions lead to the definition of class hierarchies and inheritance, concepts that are not central to structured analysis or data modeling. An object oriented analysis model is more robust and more likely to contain information that is reusable in future related problems [Shlaer 88, Coad 91]. Since reuse is of great interest to the IS industry [Fichman 92], IS students need to experience object oriented analysis.

Object oriented analysis may be taught as a

"stand-alone" excursion into object orientation.<sup>6</sup> It is unclear to what extent the underlying concepts of object orientation can be fully comprehended if the student's only introduction to the discipline is limited to this area of analysis.

## 2.3 Object Oriented Design

Because the design activity is so closely associated with implementation it is difficult to assume that it can be uncoupled from the programming environment which will be the target of the design. As discussed above the class library contributes significantly to the student's learning curve. As a foundation for the classes that will be defined for the specific program the class library represents an important encyclopedia of tools to support all sorts of things from data structures to desk top icons. Although many aspects of the vendor provided class libraries are not germane to basic education in algorithms or even programming languages, they may not be ignored because the environment in which the object oriented programming language is wrapped usually requires some of these classes if credible programming exercises are to be attempted. Indeed, interface design, interactive dialog, and icon driven program control are the trend in object oriented programming language environments and this is reflected in emerging object oriented design methodologies [Coad 92].

## 2.4 Object Oriented Database

Object oriented database may be an overly optimistic label. It is probably more correct to refer to classes to support data management in object oriented programming environments. Although graphical user interfaces are evolving that allow the manipulation and data definition of stored data in object oriented databases, the primary interface remains some object oriented programming language. ONTOS, for example, is delivered as a class library upon which the database designer builds a tailored database management system by choosing from behaviors and structures predefined within the class library. For the most part object oriented database is beyond the reach of organizations who do not have access to object

---

<sup>6</sup> Just as a structured analysis model may be used as the basis for an entity-relationship design it is possible to use an OOA model as the basis for design in a different modeling approach. One would expect a significant loss of overall cohesiveness and reusable documentation.

oriented programming specialists. Unlike relational database systems which generally provide a developer interface that is independent of host level programming language knowledge, object oriented database at this point in time is basically an extension of object oriented programming environments.

## 2.5 Object Orientation Implications for the IS Curriculum

Object orientation represents a significant new direction in essence and style for design, programming and database. Object oriented analysis may be conducted at a sufficiently abstract level as to allow it to co-reside with structured analysis and data modeling approaches. It would seem imprudent to attempt to address object oriented analysis without some automated tools support. (This is true of most approaches today!)

Object oriented database is nearer its infancy of evolution than any of the other facets. At this writing it does not enjoy the influence in the database market that would make it imperative to equip all IS graduates with a deep course experience in object oriented database.

Object oriented design and programming are timely and influential facets that must be addressed soon. They may also prove to be the most expensive from a curricular perspective as they do not neatly dovetail with existing programming and design paradigms in IS curricula.

The following section poses three alternative approaches to interjecting object orientation into the IS curriculum based on these implications.

## 3.0 Object Orientation Curriculum Strategies

This section discusses three strategies for incorporating object orientation into an IS curriculum. They range from introductory to fully integrated: horizontal, vertical, and full integration. Costs, benefits and risks are discussed for each approach.

### 3.1 A Horizontal Integration Approach

A horizontal approach to incorporating object orientation into the IS curriculum would introduce it as learning units in each of the "affected" courses. These would surely include: introductory and advanced programming, systems analysis, systems design, data management, and software engineering (under various titles and aggregations).

As learning units the topics of object orientation may be subdivided and "piggy backed" onto material closest to its relevance. In programming for example object orientation might be a section which introduces inheritance and polymorphism on the heels of abstract data types. It may be explored as an elaborate coding feature similar to macros without an in depth treatment of class libraries and GUI sensitive development. In analysis it may be treated as one of many alternative methodologies for requirements elicitation. In data management it may be introduced as an advanced evolving technology not unlike database machines and distributed database topics. In design it may be addressed as an extension of OOA and diagramming as an alternate form of systems documentation.

A basic assumption of this approach is that students receive a well rounded awareness of object orientation that would prepare them to develop significant professional skill as needed in their careers. It is unlikely, for example, that graduates would have any appreciable capability with object oriented programming or have any functional expertise in C++, Smalltalk, or Object Pascal. However, they should be able to participate as users and contributors to object oriented development projects.

### 3.2 Benefits, Costs and Risks of the Horizontal Integration Approach

The primary benefit is the minimal disruption of an existing curriculum. By layering object orientation as another perspective on system models the initial cost of tools and faculty preparation is contained. The degree of emphasis may be evolved in a distributed manner as each area becomes more pressured to adopt object orientation as an underlying model. Students receive some awareness in object orientation's breadth of influence on IS and are better prepared to recognize a specific need for training if an employer moves in the object oriented direction. This posture may allow a curriculum to sustain an "up to date" appearance while the object orientation technologies mature and become more stable and at some future time may be adopted with less risk.

Curriculum costs are limited to introduction materials rather than full laboratory preparation. There is cost in coordinating the dispersion of object orientation facets to the various courses. To gain visibility benefits it may be necessary to review and republish course descriptions, syllabi, and curriculum overviews. Some responsible agent must be identified

as coordinating the facet dispersion to maintain accountability for the integration's progress and success.

The horizontal approach risks diffusion of interest and commitment to the integration that will dilute to trivial proportions over one or two years. The naiveté of the market place may cause students to expect too much from the introductory commitment and the curriculum's credibility may suffer. The task of coordinating syllabus material in several courses with several other individuals is highly risky unless elaborate accountability and a course review mechanism are introduced along with the integration thrust.

### 3.3 A Vertical Integration Approach

A vertical integration approach to object orientation localizes the various facets of object orientation into one or two specialized courses (either required or elective). Object oriented analysis, design and programming are treated as parts of a unified life cycle paradigm and object oriented database is treated as a specialized application of class library technology.

The course(s) here are necessarily positioned in the upper tier of coursework depending on prerequisites that teach algorithms, programming, data structures, and general analysis and design. Students enter this course(s) as credible programmers and system developers. They leave with apprentice knowledge of object oriented programming in Smalltalk and (possibly) Object Pascal.

An assumption of this approach is that students who choose the specialized courses will gain a literacy level of professional skill and be positioned to advance quickly with further professional education at work or in advanced object oriented courses. They should be able to assume positions on object oriented development teams and transfer their hands on experience to whatever object oriented tools they are equipped with in the workplace. Completing this specialized coursework would allow the graduate to credibly assert competence (not mastery) in object oriented analysis, design, and implementation which would significantly enhance their resumé.

### 3.4 Benefits, Costs and Risks of the Vertical Integration Approach

A benefit of this approach is the concentration of discussion on object orientation with references to

alternate paradigms only for contrast. The faculty impacted by this approach are limited to those who are responsible for the course(s); impacts include lectures, tool training, and laboratory preparation. Controlling the student's exposure to the topic is simplified and the quality of the student's experience is more readily measured and certified for publication to the IS marketplace.

The primary costs for this approach are the preparation for the faculty who will deliver the course(s) and the acquisition and support of tools and laboratory for the students' practical experience. Because all the facets of object orientation are focused in this course(s), it may be necessary to supplement the teaching faculty with laboratory assistance for the wide range of tools that may be appropriate.

Because the faculty dedicated to the thrust are limited in number there is a greater dependency on those faculty for the longevity of the thrust. The range of automated tools that may be used will be somewhat limited by the preparation time available to the "few" faculty teaching the courses. Since all facets of object orientation are concentrated in this course(s) the range of tools may be greater than usual.

### 3.5 A Full Integration Approach

A full integration approach to object orientation supplants any other system model of development. In this approach the curriculum is restructured to focus on the object oriented concept development. Analysis, design and programming are centered on object orientation with the goal of producing a fully competent object oriented developer. Structured analysis, data modeling and other paradigms would be introduced primarily in contrast to the object oriented approach.

This approach introduces students to object orientation as their primary mind set for system modeling. Students develop expertise in Smalltalk, C++, and Object Pascal with hands on experience in object oriented database. The graduate is prepared to participate as a principal or leader of an object oriented development project.

### 3.6 Benefits, Costs and Risks of the Full Integration Approach

A benefit of this approach is a clearly unified curricular goal focused on producing developers fully professionally prepared for object oriented development. Some economy is gained by lessening the need for tools and laboratory support for

structured and non-object oriented learning tools. The majority of the faculty are indoctrinated into object orientation and significant levels of interchangeability are present. Students completing this program are highly sought after.

Reorienting and retooling the majority of a faculty is expensive (if at all possible). There are course oriented texts available for programming, some for analysis and fewer for design or database. It may be necessary to develop extensive local materials to support a full range of IS courses in an object oriented context. The tools of object orientation are expensive in acquisition, learning, and support. The choice of tools carries its own risks. As the discipline is yet rapidly evolving the stability of the existing object oriented tool set is uncertain. It may be necessary to move through several tool versions or even tool systems within the span of just a few years.

There is a significant risk that such a refocusing of an IS curriculum would outstrip the local IS marketplace's ability to absorb its graduates. The resulting "mismatch" perception could be very detrimental to the program's survivability. It is also uncertain if the majority of a faculty in a department could transition to an object orientation smoothly enough to maintain adequate student support and rapport.

A final risk is that object orientation may not become a mainstream IS application development paradigm. Although this seems unlikely, it is possible that GUI advances that further simplify the end user's access to the programming power of object orientation may bypass the C++ and Object Pascal generation of tools completely as far as IS applications are concerned. If this were to become the case, the costs of full integration associated with programming (and probably database) might be perceived as wasted.

### 3.7 An Assessment of Object Orientation Curriculum Strategies

The three strategies posed above represent only three points on a continuum of alternative approaches. They do capture a broad range of the issues that the curriculum designer must address.

It would seem that the risks posed by the full integration alternative are too great to be feasible for IS curricula at this point in the evolution of object orientation. Although object orientation offers great promise for software reuse and increased productivity it may represent too dramatic a paradigm shift. The nature of existing systems and the legacy of structured

methodologies and tools will persist for many years to come. It is unlikely that IS development shops will abandon these approaches within the first 5-10 years of a graduate's career.

The horizontal integration alternative appears most appropriate for an IS curriculum which is currently evolving an emphasis on end user systems development and tools. As such it may have a limited commitment to IS application development in favor of systems integration and office productivity enhancement. The lack of professional skill development in object orientation that this approach entails would be consistent with the end user focus of such a curriculum.

The vertical integration approach offers the greatest promise for those IS and applied computer science curricula which emphasize the graduate's ability to assume the role of system developer in an IS organization. The continued marketability of structured methodology skills, data modeling experience and pre-object oriented database technologies represents too great a benefit to be discarded (as in the full integration approach). This approach starting with a specialized course(s) allows object orientation to infiltrate other analysis, design, programming or database courses in response to the maturation of the discipline or changes in the marketplace. The risks are confined (and concentrated) in a limited area of the curriculum where they may be addressed without undue interference on other courses or faculty.

## 4.0 Summary

Organizations are already sifting through IS graduates looking for awareness, competence, and mastery of object orientation. IS curricula must react to the needs of industry, employers and students. Although the prevailing vision of object orientation at this time may be object oriented programming (caused largely by computer science's perspective), object orientation must be recognized by the IS community as more than a programming issue.

Object orientation represents a series of challenges to the IS educator. It engenders a fundamentally new system model which may impact the orientation of all courses that deal with modeling computer systems. It embodies new sub-technologies in programming, analysis, design, and data management which impact the respective coursework. It ushers in a new set of methodologies and software tools that require training, configuration, laboratories and pedagogy. It challenges our traditional structured



approach and data modeling approach for primacy in our image of the proper IS professional's mind set.

We have attempted to frame the object object orientation challenges to provide curriculum designers a starting point for crafting and evaluating their particular response to the new and exciting systems development opportunities encompassed by object orientation.

## References

- [Alabiso 88] Alabiso, B., "Transformation of Data Flow Analysis Models to Object Oriented Design," OOPSLA-88 Proceedings, September 1988, pp. 335-353.
- [Borland 92] Borland Corporation, *Turbo Pascal Version 7.0: Users Manual*, Scotts Valley, California, 1992.
- [Coad 91] Coad, P. and Yourdon, E., *Object-Oriented Analysis*, 2nd Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.
- [Coad 92] Coad, P. and Yourdon, E., *Object-Oriented Design*, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.
- [Dahl 66] Dahl, O. J. and Nygaard, K., "Simula: An Algol-Based Simulation Language," *Communications of the ACM*, Vol. 9, No. 9, September 1966, pp. 671-678.
- [Duff 86] Duff, C.B., "Designing and Efficient Language," *Byte*, Vol. 11, No. 8, August 1986, pp. 211-224.
- [Fichman 92] Fichman, R. G. and Kemerer, C. F., "Object-Oriented and Conventional Analysis and Design Methodologies," *IEEE Computer*, Vol. 25, No.10, October 1992, pp. 22-39.
- [Goldberg 83] Goldberg, A. and Robson, D., *Smalltalk 80: The Language and its Implementation*, Addison-Wesley, Reading, Massachusetts, 1983.
- [Korson 92] Korson, T. D. and Vaishnavi, V. K., "Analysis and Modeling in Software Development," *Communications of the ACM*, Vol. 35, No. 9, September, 1992, pp. 32-34.
- [Ladd 90] Ladd, S. R., *C++ Techniques & Applications*, M&T Books, Redwood City, California, 1990.
- [Moon 86] Moon, D. A., "Object-Oriented Programming in Flavors," *OOPSLA '86, SIGPLAN Notices*, ACM Inc., Vol. 21, No. 11, November 1986, pp. 1-8.
- [Shlaer 88] Shlaer, S. and Mellor, S. J., *Object-Oriented Analysis: Modeling the World in Data*, Yourdon Press, Englewood Cliffs, New Jersey, 1988.
- [Stroustrup 86] Stroustrup, Bjarne, *The C++ Programming Language*, Addison-Wesley, Reading, Massachusetts, 1986.
- [Symantec 91] Symantec Corporation, *THINK Pascal: Object-Oriented Programming Manual*, Cupertino California, 1991.
- [Tesler 85] Tesler, L., "Object Pascal Report," *Structured Language World*, Vol. 9, No. 3, 1985, pp. 10-14.
- [Ward 89] Ward, P. T., "How to Integrate Object Orientation with Structured Analysis and Design," *IEEE Software*, March 1989, pp. 74-82.

## GROUP SYSTEMS IN THE CIS/MIS CURRICULUM

### Panelists

Evangeline Jacobs, Lecturer in Computer Information Systems  
Northern Arizona University

Robert Lynch, Associate Dean, College of Business Administration  
University of Northern Colorado

Douglas Vogel, Associate Professor of Management Information Systems  
University of Arizona

### ABSTRACT

During the past year, Group Systems have received much attention in the popular press and several vendors offer products to support group efforts. This panel will explore the role and use of Group Systems in the CIS/MIS curriculum for instruction, service, and research. Panelists represent schools that are Research Universities for Groups Systems V.

Panelists represent three viewpoints. The University of Arizona is a pioneer in the development of Group Systems. They currently have three group facilities varying in size from 40 stations to 14 stations. The facilities are used for instruction, community, university, and college meetings, and are available for use by outside government and business organizations. In addition, UA is actively involved in research and development of Group Systems and outreach/training in Group Systems worldwide.

The University of Northern Colorado has had a Group Systems facility for three years. Their 22-station facility is used for class sessions and university and college meetings, and is available for use by government and business organizations. UNC has both faculty and graduate student facilitators and offers a short training course for facilitators each fall.

Northern Arizona University established a 14-station Group System facility early this year. They are in a start-up phase of training facilitators, interesting university and college committees in using the system for meetings, and introducing group support modules into courses.

The panelists will discuss how group systems are used for instruction, research and service within their academic environments. Specific issues that will be discussed include the following:

Is the Group System used across disciplines?

Are Group Systems used for administering the department, college, and/or university?

How are facilitators selected, trained, evaluated, and rewarded?

How is the Group System facility funded?

What types of group support activities are groups using the facility for?

What research projects are currently underway using Group Systems?

What courses include group support as part of their curriculum?

Following the presentation by each panelist, the panel will welcome comments and questions from session attendees.

## The Future of the MIS Course

Raymond McLeod, Jr., Texas A & M University  
Richard A. Hatch, San Diego State University  
James A. O'Brien, Northern Arizona University  
James Senn, Georgia State University  
Mary Sumner, Southern Illinois University at Edwardsville

### Abstract

The MIS course has traditionally had the responsibility for describing computer use in business to business students. As that computer use has changed over the years, the course has been forced to change as well. The authors describe the origin of the MIS course, identify several key questions that must be answered in a changing business environment, present a conceptual framework for classifying expected future changes, and identify areas of emphasis for achieving a successful MIS course.

### The Origin of the MIS Course

The MIS concept originated in the Mid-60s and was immediately embraced by both industry and academia. Colleges began to add MIS courses in the late 1960s and by the early 1970s the practice had become widespread. The reasoning was simple. It was clear that computing was an emerging business discipline, but little was known about how the computer could be applied outside the accounting area.

Unlike today, there were not many MIS textbooks from which to choose. The real pioneers were Robert G. Murdick and Joel E. Ross of Florida Atlantic University. [3] Their text featured a management orientation that set the tone for many MIS authors who followed. Also, during those early years, faculty were blessed with two excellent MIS case books, both published by Irwin. One was authored by John Dearden, F. Warren McFarlan, and William Zani, all of Harvard [1], and the other by Thomas R. Prince, of Northwestern [4]. Both books contained meaty cases involving computer use in real organiza-

tions, and helped educate business students to the fact that computers could be used by managers to solve problems.

In order to appreciate the importance of the MIS course during its early years you have to understand the pathetic lack of knowledge of business computing. In the firms, the computing equipment was closely guarded by the MIS departments. Users were given tours, but otherwise were not allowed access to the equipment. In the college classrooms, the students suffered the influence of the computer vendors who emphasized computing technology rather than application. In short, very few people understood how computers could be used in business. The original intent of the MIS course was to overcome this lack of understanding.

### Changes in the Business Environment

Some of the major challenges and issues in teaching the MIS course result from changes in the business environment. Today,

managers are faced with downsizing, reengineering, TQM, strategic relationships, and major technological advances.

Many organizations are becoming "information based." Companies, like Mutual Benefit Life, are creating new jobs such as "customer service representative." With access to powerful desktop workstations, customer service reps can search databases for information on policies, claims, and policyholders. At IBM Credit, "deal structurers" use microcomputer-based workstations to authorize credit arrangements. In both cases, the use of information technology (IT) is critical to designing new jobs and providing individuals in those jobs with greater autonomy in decision making.

The MIS course is designed for current or future business managers and should provide them with insight into the concepts, tools, and applications that will enable them to answer the following questions:

**What information do I need?** This seemingly simple, yet extremely difficult, question is complicated by changing business conditions, industry competition, and internal priorities. In the MIS course, students need to explore methods of determining their information needs. These methods include concepts such as critical success factors, methodologies such as BSP, and techniques such as Ends/Means Analysis.

**What new technologies should we pursue?** Managers need an understanding of technology and its applications. Which option should be pursued: LANs, CASE, DTP, client-server, and so on? Technology mistakes can be hazardous, and to minimize the

chance of that happening, the manager needs to understand the importance of standards in communications, database, and software design.

**What kinds of information systems support business needs?** Managers will be using information systems to support decisions within functional areas, such as sales and marketing, accounting and finance, production, and human resources. These functional information systems support day-to-day operations, tactical decisions, and strategic planning.

**How do we develop information systems plans?** The issue of what information systems enable the firm to compete most effectively is a basic premise of information systems plans. One of the key issues in the era of strategic information systems is whether IT can provide organizations with a competitive advantage. Rather than playing "follow the leader," a firm can gain a long-lasting advantage by focusing on information systems that reinforce capabilities-based competition. This means using IT to streamline key business processes such as customer service. A good example of how this can be achieved is the Wal-Mart inventory system for Pampers that relies on the manufacturer, Procter and Gamble, to maintain inventory at the proper level.

**What is the manager's role in information systems design?** In today's organizations that feature decentralized IT resources, managers play a critical role in analyzing current business activities and designing new information systems. Reengineering fundamental processes is an essential part of the process of systems analysis. A number of approaches in information systems

design, such as RAD and prototyping, give managers an opportunity to define and refine their information requirements as the systems are being built.

**What systems design options are most appropriate?** Each year the range of design options increases to include 4GLs, prototyping, a multitude of software packages, inhouse development, outsourcing, and end-user development. The key to success is selecting the appropriate option based on the characteristics of the application.

**What are the responsibilities of MIS management?** MIS management has a diverse set of responsibilities, such as technology planning, user-support, management of databases and network projects, and telecommunications systems management. Today's MIS professionals assume more of a consulting and facilitating role in relation to these responsibilities.

The sum total of this changing business environment is the requirement that the MIS course focus on the manager's view of technology, applications, and IS design. In addition, hands-on application development will become an increasingly important element of the course. The theory, concepts, and applications of systems design will be built into the course as well.

### **A Conceptual Framework**

Another way to organize our thoughts about the IS knowledge needed by managers, and, therefore, the content of the MIS course, is a framework consisting of five areas: (1) foundation concepts, (2) technology, (3) applications, (4) development, and (5) management.

**Foundation Concepts** The MIS course has typically included a healthy dose of behavioral concepts from organization theory, management theory, and problem solving and decision theory, and more technical concepts from general systems theory, information theory, and information technology. In the future, we shall see more emphasis on behavioral and technical concepts from a variety of disciplines that underlie topics such as workgroup team building, decision making and productivity, business process reengineering (BPR), competitive advantage, technology absorption, and ethical issues.

**Technology** In an MIS course the "technology" typically means a discussion of hardware, software, telecommunications, and database management concepts and developments. We will probably see more emphasis in the 90s on the technology needed to interconnect and enhance the productivity of end users, workgroups, organizations, and business processes. Examples include client/server computing networks, cooperative work software, object-oriented software and databases, and using AI techniques such as expert systems and neural nets to enhance operational processes and end-user productivity and decision making.

**Applications** IT is applied to business in a wide variety of applications to support business operations, managerial decision making, and strategic advantage. MIS courses have therefore traditionally spent much time discussing applications such as transaction processing, OA, and DSS. However, applications such as end-user and workgroup productivity, strategic and interorganizational systems, BPR, and some AI applications have gained recognition as topics that should

be emphasized in the MIS courses of the 90s.

**Development** Managers must know how to develop IS solutions to business problems. We no longer expect business students to know the details of programming and programming languages. Even the details and techniques of the traditional SLC or CASE methodologies are no longer necessary. MIS courses will now emphasize end-user development. That is, how end users can use a variety of software packages to prototype and implement application systems for themselves and their workgroups, with or without the help of IS professionals.

**Management** Information systems and IT must be managed by managers from the departmental level to the CEO. Whether called IS management, IT management, or information resources management, this topic has gained increased emphasis in MIS courses with the recognition that IS/IT can have a major impact on the success or failure of a business. Examples of topics gaining increased emphasis include managing end-user computing and development, managing technology absorption, managing IT for competitive advantage, and global IT management.

#### **Areas of Emphasis**

The most successful MIS courses, across a broad range of institutions, are distinguished by their areas of emphasis, which transcend choice of course materials and delivery methods. The four areas of emphasis include: (1) IT capabilities, (2) applications influence, (3) problem solving, and (4) player's roles.

**IT Capabilities** At one time, the principle emphasis was on the characteristics of the technology itself: details of computer processing circuitry, data representation, and input/output/storage devices. Today's successful MIS course, on the other hand, emphasizes capabilities, not capacities. It explores the manner in which IT can be applied in assisting individuals and organizations to be both productive and effective. IT is still important, but only within the context of broader IS functions, ranging from capture to transmission, and discussion of their importance to business.

**Applications and Organization Success** Early MIS courses devoted large amounts of time to definitional issues, such as the distinction between data and information, and between DP, MIS, and DSS. Today's successful MIS course focuses on operational systems, sometimes called mission critical or life stream systems, as well as management-oriented applications, with meaningful exploration of the link between each.

Of particular interest are the strategic uses of IS. Here, it is important to recognize that the term "strategic information system" does not pertain to a category of information systems, but rather to the way in which systems are used. Applications discussions are also most useful and complete when they recognize that IT has a role in the extended organization. Hence, both intraorganizational and interorganizational systems are an integral part of the course, not as a topic, but as a theme running throughout the course.

**Problem Solving** The ability to evaluate a business situation and determine the potential impact of IT, whether favorable or detrimental, is a fundamental capability in organizations. It is not limited to IT professionals. Identifying and solving problems is a fundamental part of business and thus a central theme in the MIS course.

Identifying the stages of problem solving provides a good backdrop against which to describe the manner in which the different facets of IT support the problem-solving process. Increasingly, BPR should be linked to problem solving, with IT providing pivotal capabilities for rethinking how problems are handled and processes carried out.

**Roles and Responsibilities of Individuals** Recognizing that students and practitioners often have the mistaken perception that IT is the responsibility of the IS specialist, MIS instructors have the opportunity, indeed the obligation, to illustrate the breadth of IS responsibility in an organization. A meaningful way of describing key roles includes the following:

**User**--both the hands-on user, who operates or directly interacts with the system, and the indirect user, who relies on information produced by an application.

**Developer**--anyone responsible for creating information systems, including those responsible for computing, software development, data management, and network management.

**IT manager**--the corporate official, by whatever name the role is called in a particular organization, who has direct responsi-

bility for overseeing the organization's IT investment.

**Functional manager**--the person who oversees investment in development or use of functional systems and has organizational responsibility for control of IS activities within the bounds of a functional area, or more appropriately, a business process.

**Senior executive**--the person who incorporates competitive and strategic uses of IS with corporate plans and strategies, as well as spells out corporate exposure to risk resulting from IS failure.

Keeping these areas of emphasis in mind, an instructor can organize an effective MIS course in many different ways. The course will be suitable for current and emerging managers who need to understand the potential contribution of IT within organizations, and enable them to demonstrate and express this understanding in an effective manner.

### **Today's Challenge**

The MIS course is facing a period of unprecedented opportunity. In corporations throughout the world, end-users are suddenly empowered to create MIS solutions. Managers can now hire their own departmental and work-group systems development from outside the corporation. Sophisticated end users are building useful personal systems on their own.

A decade ago, users turned to the MIS department for solutions, and they needed to know primarily how to request IS services effectively. To profit from today's empowerment, users at all levels need practical, down-to-earth information about



how to increase their own MIS productivity. That is our opportunity.

An interesting symbol of this change is the Personal/370 Adapter card for PS/2 computers now sold by IBM to OEMs. It runs both the VM and VSE mainframe operating systems--soon MVS--on the PS/2. The card is marketed as "ideal for corporate departments that want to downsize but haven't because of the trouble and expense of porting mainframe applications that are shared by a small number of users. [2]

The opportunity of emphasizing ways for users to increase their MIS productivity, however, carries with it an accompanying risk. In higher education, some of our colleagues are drawing the wrong conclusion. They say "Corporations are cutting their MIS departments, and we should too. Besides, today's students already know how to use spreadsheets, so they don't need an MIS course." These arguments may be advanced especially vigorously at schools where business enrollments are dropping and resource allocations of all kinds are being reevaluated.

We who teach MIS must respond to these mistaken assumptions vigorously and forcefully. As MIS responsibility shifts from MIS departments to users, the need for broad-based MIS education goes up, not down. Our business school colleagues need to see this logic.

One step in the right direction is to ensure that our MIS courses reflect the new reality. For example, a course with an obvious mainframe and MIS department orientation that would have been appropriate a decade ago is not only drifting out of date,

but also drifting into danger. Such a course will be difficult to defend when resources contract and responsiveness to customer needs becomes the criterion.

### Bibliography

[1] Dearden, John; F. Warren McFarlan; and William M. Zani, Managing Computer-Based Information Systems (Homewood, IL: Richard D. Irwin, 1971).

[2] Fisher, Susan E., "PS/2 May Gain Mainframe Card," PC Week August 2, 1993, p. 67.

[3] Murdick, Robert G., and Joel E. Ross, Information Systems for Modern Management (Englewood Cliffs, NJ: Prentice-Hall, 1971).

[4] Prince, Thomas R., Information Systems for Management Planning and Control (Homewood, IL: Richard D. Irwin, 1966).

## PROFESSIONAL PRACTICE: NETWORKING WITH BUSINESS

PANEL DISCUSSION  
WARREN DUCLOS, TULANE UNIVERSITY  
AND EDSIG BOARD MEMBERS

### Abstract

To position the EDSIG session at ISECON, the following draft of a "Source Book for Student Professional Practice" is included in the Proceedings. In many academic programs in business areas and in applied arts and sciences, a professional practice component is incorporated into the overall curriculum plan. While this is not a document urging the development of cooperative education programs universally, it suggests that there are a variety of ways to improve networking between postsecondary education and business -- and one of these is through professional practice experiences.

---

EDSIG SOURCE BOOK FOR STUDENT  
PROFESSIONAL PRACTICE IN  
UNDERGRADUATE COMPUTER  
INFORMATION SYSTEMS PROGRAMS  
(A DRAFT)

### Introduction

Increasingly prevalent throughout postsecondary education is the value being placed on a "professional practice" component -- giving students a guided experience in the "real world" of work before completing their undergraduate degree. Such professional practice includes different types of learning experiences in the work place.

The experiences vary across campuses in the extent of responsibility expected of the student, the time the student spends on it, and the amount of new learning encountered.

In different college or university programs, the degree of emphasis on such professional practice components may vary widely. In some cases it is optional and in others required; in some it lasts for a week or two of one academic term, in others a full term or longer.

EDSIG completed its Faculty Advisor Network (FAN) survey in 1991. Respondents clearly indicated the need for EDSIG to provide normative information and sample materials in the area of professional practice.

(Reported for the EDSIG Board by Kathryn McCubbin in Journal of Information Systems Education, Fall 1991, Volume 3, Number 2, pages 37-41.)

Note: A summary of the "write-in" responses pertaining to professional practice will be included in an Appendix of the final draft.

EDSIG offers this draft of a Source Book to all institutions with undergraduate programs in Computer Information Systems, whether they are starting or improving a professional practice program. Ideas and materials have been provided by contacts at ten colleges and universities in the hope

that others might find them informative and useful.

Participants in this project, who provided information and/or materials:

California State  
University -- Pomona  
Illinois State University  
Loyola University of  
Chicago  
Ohio State University  
Purdue University  
Southwestern Michigan  
College  
Tarleton State University  
Tulane University  
University of Wisconsin  
-- Whitewater  
Winston-Salem State  
University.

#### Some Existing Models

A number of models of professional practice are utilized in Computer Information Systems programs. Three major examples are cooperative education, internships, and outreach.

a. cooperative education -- a full term, alternating terms, or one year; full-time work program and work is related to academic study; required for the academic degree;

b. internship -- usually over a single term; some work and some academic study; may be incorporated within a course or seminar, under the guidance of a professor;

c. outreach -- a less ambitious but coordinated exposure; any short-term experience; could entail a series of site visits and discussions, or 2 to 3 days in some business environment(s),

or a modular practicum within an academic course or seminar. (The models will be more thoroughly specified in the final draft of the Source Book; sample college programs will be included.)

#### Values and Advantages

The varieties of professional practice models are rooted in different underlying values and therefore pursue different goals and objectives.

Values/Rationale. Some models aim primarily to provide a service to students, over and above those attained in the classroom or lab. Others seek to increase or improve collaboration with corporations and agencies. Finally, professional practice programs are used as an additional public relations and recruiting tool for a college or university program by increasing its reputation in the community at large.

Goals/Benefits. Goals and benefits of programs in professional practice are typically set to attain a variety of outcomes.

First, for the student, the real world experience itself is invaluable. Included is the technical experience that is gained in the use of new or advanced hardware or software. Others are often more important.

Professional practice programs provide an opportunity to develop work environment skills, such as proper dress, punctuality, time-management, team/group dynamics, professional work habits, communication and

interpersonal skills. Students who have these experiences are more prepared to move from academe to the work place and are often more likely to be hired. Sometimes, academic credit can be attained. Sometimes, salary and/or benefits are associated with the experience. Students can adjust the remainder of an academic program (e.g., electives), in accord with insights gained in the experience, a clearer picture of what awaits them upon graduation and for which they can prepare. The experience can often be a source of maturing and motivation for languishing students or a reward for honors students; it builds new connections for curriculum courses and gives new meaning to academic work.

Second, there are goals and benefits for business and industry, such as publicity and exposure gained by performing a service to the community via the college or university. Students return to the university and serve as the company's good-will ambassadors. Although no commitment is made, the professional practice program can be used to supplement the company's employee recruitment efforts, resulting in decreased costs for drawing entry-level workers to the company. There can be a reduced time and expense to initiate students hired, since students become productive earlier in their formal employment with the company. The program can be used as a phasing or probationary period for prospective new employees, providing a jump start for

company training. Throughout this time period, moreover, students are in a position to offer new ideas and approaches to the work being done in the company.

Third, there are goals and benefits for the academic program and institution. Feedback from business and industry experience will provide indicators and directions to improve the academic program. Insights and ideas brought back from the "real world" provide an opportunity to enhance or supplement the curriculum. A professional practice component provides increased visibility or publicity to the academic program, making it more attractive to prospective new students. It increases the opportunity for general public recognition and is a demonstration of an institution's cooperation with business and industry.

### Responsibilities

An academic unit considering improvement or initialization of a professional practice program must clarify the responsibilities of all parties participating in the program.

1. Of the student. Each student participant must exercise a commitment to learning in a professional environment. There must be an agreement to follow the University's and the employer's rules, policies and procedures. Students complete all required administrative paper work and registers for university credit, if

applicable. They must complete any term and project begun in a professional practice experience -- or withdraw in accord with standard procedures. They must meet all obligations in a professional manner during the period of the experience, especially any required tasks for the company. Finally, they must cooperate in any university effort to assess the employer and the work experience.

2. Of the business. The cooperating business must make and fulfill a commitment to provide professional practice experiences that are typical of the enterprise. It must provide supervision of the student as agreed upon with the university. It must make clear the prospects of employment for the student after graduation and the conditions on which employment might be offered. It must submit to the university a job description for the experience and provide the student the professional experience for the agreed upon period of time. If the corporation decides to change the program of a particular student, the corporate coordinator will inform the university coordinator. Evaluation of students' work during and on completion of the professional practice experience is a requirement; the corporate coordinator provides feedback to the university coordinator as to the student's performance and the workability of the program.

3. Of the academic unit. Overall administration of the program remains the

responsibility of the department, college or university. This includes supervision of the professional practice program on the student side, including application and eligibility process, coordination of placement, periodic checkins, and other support structures. Also included are appropriate controls in selecting sites for professional experiences of students, as well as receiving feedback and performing overall program evaluation.

#### Some Issues to Resolve

When planning to initiate or enhance a professional practice program, program developers will need to make decisions on several key issues. This list is intended to be representative, not complete.

1. What are the eligibility requirements? Is the experience required, optional or selective? If selective, what are the criteria? Will the student be required to have completed a number of credits or to have achieved a minimum overall or major grade point average? Is the experience available to all? Are letters of reference required?

2. Can academic credit be earned or not? If yes, how is the amount of credit to be determined? Will there be an allowable maximum of academic credits? Will the number of credits be connected to the number of hours worked? Or will the number of credits be tied to the degree or amount of new learning experienced

(or other factors)?

3. If credits are earned, how will a student grade be assigned? Will the evaluation for a grade be made by the corporate coordinator, by the University coordinator, by someone else, or by a combination of parties?

4. What is the placement process? How are final site selections and student assignments made?

5. Are wages and/or benefits involved? If so, which and how much?

6. At what locations or sites will professional practice experiences be available (specific businesses, governmental agencies, non-profit groups, schools, charitable organizations, utilities)?

7. In each case, will the experience involve a job-entry path or not? Is the company or agency actually hiring new employees now?

8. What is the timeframe for the experience (one term; multi-term; partial term; Summer only; one full year)?

9. What will be the student's work schedule (number of hours per day, days per week, weeks per term)?

10. What will be the scope and range of experience during the professional practice program (a broad overview; some varied tasks on specific projects; fewer, but complete project assignments)? (I.e., will the student be required to do a little of everything or to complete one project for one department?)

11. What will be the degree of structure? E.g., will there be a contract signed by all parties, a

specified list of required experiences to be provided, or some other (these depend on the program's overall values and model?)

12. Where will administration of the professional practice program reside (i.e, in what office within the institution -- in a placement office within a single college, in a central placement office of the university, or in the information systems department itself)?

13. What are the costs to the institution and who pays?

14. Will the program's Public Advisory Committee and/or the Alumni Association be involved? How?

15. How will all parties work to maintain and integrate the student's focus on the academic program (tie the professional practice experiences to the academic program and its objectives)?

16. What will be evaluated -- the student and the work performed, or the company and the support provided, or the academic department and the direction given by it? How and by whom will evaluation be performed?

### Support Structure

Whatever the model and scope of the professional practice program, the academic department and institution will have to provide some sort of support structure for the program. The amount and extent of structure will vary, depending on the model and goals adopted. This section includes some examples.

1. Preliminary Support. The process begins with the identification by the institution of business "hosts"; the school must look for willingness to cooperate, an appropriate technical environment, and a corporate contact at each site. It is desirable to develop a pool of sites, together with contacts and coordinators at each site.

Other initial support components include: determination and publication of eligibility criteria for student participants; identification of in-house mentors and coordinators; definition of the application and matching process for students with corporate sites; definition and development of information systems to support the program; resolution of all items in the previous Section of this Source Book.

2. Application and Assignment. The institution and cooperating corporations work through the selection process for students; available sites are established and publicized; the placement of students within current openings is accomplished.

3. In-Course. The institution's coordinator or field directors perform checkins and supervisory site visits, to make sure assignments are clear and to review and communicate expectations.

Interim reports and controls are sometimes used. Some programs require students to maintain a log or journal of the experience; feedback on problems is communicated and resolved.

4. Completion. Final reports are collected from the student and from the business site coordinator. Individual performance evaluation is effected. Overall program evaluation is made and recommendations for improvement are prepared and delivered, as are any other items needed to achieve closure. A post-term review may be made by all parties involved in the program.

5. If the experience takes place simply as a module of a single course or seminar, guidelines must be developed and distributed to faculty regarding: work requirements, standards of work procedures and products, and grading.

#### Evaluation/Assessment

No program will be successful without an evaluation component. Evaluation should be oriented to both product and process -- to the outcomes of the experience and to the ways and means by which the program worked during the term. How close did we come to achieving what was defined as desired at the outset of the program experience? How well did we go about achieving it?

1. By the student. Students evaluate a variety of concerns, for example, the content of the experience: what work was done and using what hardware and software? What was actually produced? in what functional application area(s) of the business?

How demanding was the work? How clear were directions from the corporate coordinator and/or supervisor?

How helpful and receptive were coworkers? How helpful was the work experience in the student's understanding of future career directions and needs?

How supportive was the university coordinator during the professional practice experience? Suggestions? How could the academic curriculum be improved, in light of what has been seen in this real-world professional practice experience?

### 2. By the Business.

Corporate professionals must evaluate the experience, too. Information might include identification data: the evaluator's position, division or department of the company, and the work environment and assigned duties.

The corporate contact will typically rate the student in (some of) these areas: technical skills; communication skills -- written, oral; problem solving; ability to work independently; ability to work in a team; punctuality; dress; ability to learn new things; ability to perform under pressure; ability to follow instructions; basic good judgement; cooperation with coworkers; mature attitude; and others considered important for the experience.

Other items might be included in this effort. How well were you able to support the student? How did the student rate in view of your expectations? Do you have any suggestions as to how the academic program might be improved, in light of your experience with our students?

### 3. By the University's

Coordinator. The institution will be concerned with receiving the evaluations of both student and corporation. But it will also attend to the satisfaction levels of both students and corporate contacts with the experience and with the program itself. Was the student challenged and motivated to make this a worthwhile experience? Was there apparent new learning for the student? Was the experience integrated with the academic curriculum?

It will also attend to criticisms of the university and its academic program or its management of the professional practice program, as well as to any need for action on the part of the university.

### Formal Program Products

The academic department should develop a program handbook for students and corporate contacts. It will include a statement of responsibilities of each party (above: student, corporate host site, university).

If a contract is required, the department will include the approved format in the handbook. Ideally, the contract becomes the basis of evaluation; the concern here is that the program cannot be successful if one party thinks a professional practice experience is one thing and another party thinks it's something else. The contract makes clear the expectations and responsibilities of all.

All standards and procedures for the program must be detailed in the



handbook. Various forms that are use throughout the experience are also included.

Here is a sample outline for such a Handbook:

1. Goals and objectives; benefits and advantages
2. Policies and procedures
3. Responsibilities of each party
4. Forms and reports, including contract, if applicable
5. Sample corporate hosts and student experiences
6. Sample calendar of events -- from initial application to final evaluation.

Some sample forms and other items from cooperating institutions -- which might be used in a program hand book -- will be included in the final draft of this Source Book.

## Conclusions

Developing or enhancing a professional practice component in a Computer Information Systems program will provide a service to students, as well as to business and industry and the academic program itself. Depending on the conceptual model adopted, the professional practice program will present a variable degree of effort and resource expenditure on the part of each institution.

Not all the suggestions presented here will be necessary in every college or university program. They are offered only as possible bases to cover in the planning and implementation processes. EDSIG hopes that the academic and business communities will find them useful, as ways to improve networking between our sectors.

## OBJECT-ORIENTED: "IT DOESN'T HAVE TO BE DIFFICULT"

SANDRA POINDEXTER                      R. BHARATH  
E-mail: fasp@NMUMUS.BITNET      E-Mail: farb@NMUMUS.BITNET  
Management, Marketing, & CIS Dept.  
Northern Michigan University  
Marquette, MI 49855  
906-227-2605

### ABSTRACT

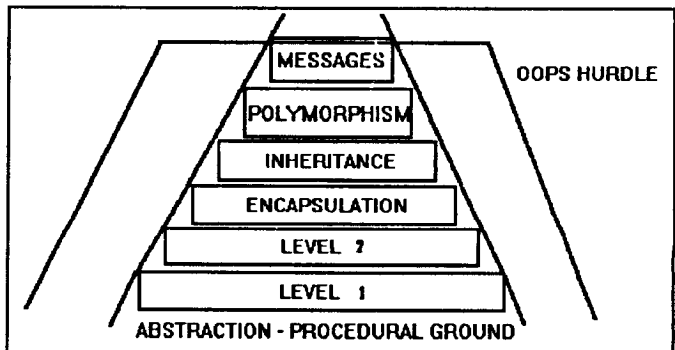
The aim of the paper is to offer a "step-ladder" to aid in climbing the object-oriented hurdle. By so doing we wish to suggest that object-oriented programming can be viewed in a way where it is seen not as a replacement, but as a generalization, of ideas which already permeate the thinking of most programmers.

Object-oriented programming (OOP), or at least the name "O-O", has gained recognition from both academia and industry as a real force demanding to be acknowledged. Reported to be an extremely useful technique for both research and business, the number and the variety of applications is increasing day by day. Nevertheless, it is true to say that there is a cloud of confusion and mistrust slowing further advancement.

O-O arose from an evolutionary process so by definition something has to be familiar to which we can cling. But, also by definition, something has to be new which we must adopt. By retracing this evolution it may be easier to differentiate between new thoughts, new terms for old thoughts, and, discarded thoughts.

By viewing OOP as a step-by-step generalization of classical data processing it may be easier to loft the hurdle. Not everything from the traditional paradigms has been discarded, and experience in the past is always useful for tomorrow. Perhaps in regard to the central concepts of objects, classes, inheritance, polymorphism and message passing, it would, be easier for people strongly anchored in the

mainstream tradition to see what the generalization provides. It allows economy of coding, with the attendant advantages of easier maintainability of large programs, and the ability to extend the use of good code by reusing it with small modifications for similar, but not identical "level 1" objects. Its costs are slower run time and higher memory requirements, both remedied by more powerful processors with large data storage devices. As hardware costs are less than software development costs, the overall benefits can be substantial. This paper has tried to suggest that, in an important sense, OOP is an extension and elaboration of habits of thought and analysis which are already part of the frame of mind of classical programming, rather than something which starts from scratch.



The Ladder

**OBJECT ORIENTATION: THE LOSS OF CERTAINTY--  
LESSONS LEARNED WHEN TEACHING OBJECT ORIENTATION IN THE COLLEGE  
OF BUSINESS INFORMATION SYSTEMS CLASS**

Wita Wojtkowski  
W. Gregory Wojtkowski  
Emerson Maxson

College of Business  
Computer Information Systems and Production Management  
Boise State University  
1910 University Drive  
Boise Idaho 83725  
E-mail: RISWOJT2@COBFAC.IDBSU.EDU

**ABSTRACT**

Many business enterprises already actively moved to or are exploring ways of migrating to object oriented environments. In these circumstances teaching object orientation becomes an imperative for proactive and forward looking Computer Information Systems curricula in the Schools of Business. In this paper we report on our first experience in teaching Object-Oriented Systems Development course. This course was constructed as a stand-alone course in the traditional CIS curriculum in School of Business. Its objectives were:

- \* Describe principles of the Object-Orientation (OO)
- \* Overview Object-Oriented Analysis (OOA)
- \* Overview Object-Oriented Design (OOD)
- \* Introduce basic notions of the Object-Oriented Programming (OOP)
- \* Overview applications of the Object-Oriented approach

We conducted our class instruction as a team endeavor. The task of examining, even in brief, subjects listed above seemed to naturally call for a team teaching approach. We broke the effort along the lines of OOA, OOD and OOP, with one person responsible for each of these areas.

From observation of our students we note that prior experience with traditional systems analysis and design might affect performance in applying object orientation. We also noted that those who had more extensive experience in building application systems were hindered by their experience and habit to much greater degree than inexperienced individuals. In this paper we ponder the lessons learned and put forth practical proposals for presenting this subject in the classroom.

# ARE SCRIPTING LANGUAGES SUITABLE FOR TEACHING OBJECT-ORIENTED PROGRAMMING?

Jo-Mae B. Maris  
Northern Arizona University

## ABSTRACT

Finding a language for teaching object-oriented programming is a problem. Scripting languages have been called "object-oriented" and might provide a solution to the problem. However, scripting languages lack some features associated with object-oriented languages but could be used to initiate students into an object-oriented way of thinking.

## INTRODUCTION

### Introduction

Get ready. COOL is coming. COOL is COBOL Object-Orient Language. COOL's arrive may not be eminent, but it is coming.

"Penetration of OOP into the commercial data processing world will take longer because the new techniques must overcome the inertia of large investments in the more familiar traditional techniques of procedural COBOL, separate data bases, and the discipline of structured development" [Howard 19].

This delay is fortunate because it gives us time to learn OOP and its application to databases. However, Howard has an additional warning, "Let's not sit idly by. Rather, let's reach into the future and control the OOP revolution. Otherwise it will control us" [Howard 19].

Teaching object-oriented programming (OOP) presents several challenges. Among those challenges is deciding what language to use. Languages such as C++, Object Pascal, SmallTalk, LISP/CLOS, Eiffel, and Oberon II are vying for supremacy [Terry]. Each language has its advantages and disadvantages [Lewis]. However, all of these languages rely on text to construct programs. Is there a choice that is more

friendly? Is there an OOP equivalent of Logo<sup>1</sup>?

One group of candidates for a friendly object-oriented language consists of authoring packages, such as HyperCard and ToolBook. A person skilled in the use of text based programming languages might ask, "Why bother considering an authoring package for teaching object-oriented programming? After all, authoring packages aren't OOP languages." Milet and Harvey express a different view of authoring packages. They state, "an understanding of hypermedia can help in the understanding of OOPs" [Milet and Harvey 2]. Milet and Harvey are not alone in their view of authoring packages. Vendors [Buchmiller], courseware developer [Brader], and BYTE [Wood] all refer to the authoring packages (ToolBook or HyperCard) as object-oriented development languages.

One may need to take a closer look at these claims about authoring packages being object-oriented programming languages, because object-oriented programming may be a victim of its own success. When a software vendor wants to sell a new development product, the product is described as "object-oriented." An exam-

---

1. Logo was created by Seymour Papert as a language that would make learning mathematics comfortable for children [Friedman 11].

ple of "object-oriented" being used by a vendor is seen in the Using OpenScript reference manual for ToolBook by Asymetrix.

"ToolBook is an object-oriented development environment that provides graphical drawing tools for creating objects and a full-featured object-oriented programming language called OpenScript for programming objects' behavior" [Buchmiller 4].

Asymetrix is not the only company to make claims of this nature. Similar statements are made about HyperCard [Brader 12]. However, this paper will deal with ToolBook.

Who is right? What constitutes an object-oriented language? What challenges are we facing when we tackle the job of teaching object-oriented programming to our students? Does a language have to use text to define all elements of a program to be a suitable tool for teaching OOP? The goal of this paper is to shed some light on these questions.

## **Purpose**

The claims such as those made by Asymetrix and Brader pose a question: Do scripting languages, similar to OpenScript and HyperTalk, provide real object-oriented programming languages? The purpose of this paper is to investigate what properties of scripting languages are object-oriented, not to censure any vendor's claims. When the vendors make claims about object-oriented programming they may be operating under different perspective of what constitutes object-oriented programming.

The possibility that scripting languages provide real object-oriented programming features is of particular interest to teachers in search of a non-threatening way to introduce students to object-oriented programming. Object-oriented programming with classes, meta classes, objects, instan-

tiation, messages, inheritance, and polymorphism can be intimidating. Therefore, if an "object-oriented" equivalent of Logo exists, the language could be useful for introducing object-oriented programming concepts.

## **Problem**

This paper investigated one candidate scripting language: "OpenScript." The problem was to determine whether OpenScript is a suitable tool for teaching object-oriented programming. Possible outcomes were: OpenScript is a suitable tool for teaching OOP; OpenScript lacks some features of an OOP language but could be a useful addition for conveying particular aspects of OOP, or OpenScript contains no valuable elements of OOP and is not a suitable tool for teaching OOP.

Since this paper will not attempt to compare several products, the following references may be useful. "An Object-Oriented Show and Tell" by Chris Terry in the June 6, 1991 issue of EDN presents several text based object-oriented programming languages and development environments, such as Eiffel, C++, and Small-Talk. "Script Languages: The BASIC of the 1990s?" by Lamont Wood appears in the April 1991 issue of BYTE. Wood presents a hacker's view of graphics based development packages, such as ToolBook, HyperCard, and Spinnaker.

## **Method**

The method used in this study consists of four parts: (1) determining criteria for what constitutes an OOP language, (2) evaluating OpenScript's properties in terms of the OOP language criteria, (3) identifying some of the challenges of teaching OOP, and (4) assessing the evaluation of OpenScript's properties in terms of the challenges of teaching OOP.

## CRITERIA FOR OOP LANGUAGES

A variety of sources agree an object-oriented language should support the following elements of OOP: classes, objects, methods, inheritance, and messages. Some examples are:

"OOP techniques draw their power from five unique concepts: Objects, Messages, Methods, Classes, and Inheritance" [Howard 14].

"The following considerations apply to implementing an object-oriented design in an object-oriented language: class definitions, creating objects, calling operations, using inheritance, and implementing associations" [Rumbaugh et. al. 297].

"The five primary concepts which distinguish object-oriented programming systems from other traditional systems are objects, methods, messages, classes, and inheritance" [Milet and Harvey 3].

Booch conveys the same idea in slightly different words:

"Object-oriented programming is a method of implementation in which programs are organized as cooperative collections of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships" [Booch 36].

Although Booch's words are different, the meaning is essential the same. For Booch, objects have state, behavior, and identity. Behaviors in Booch's model are the same as methods in other models. For Booch cooperation and behaviors incorporate the concept of communication within and between objects. Messages are just

one implementation of communications. Classes and inheritance are directly stated in Booch's synopsis of OOP. Thus, the consensus elements of an object-oriented language are: classes, objects, methods, inheritance, and messages. These model elements will be considered in more detail.

### Objects Are Building Blocks

For a language to support objects, the basic building block of the language should be objects. Object-oriented programming uses objects rather than algorithms as the basic building blocks. When algorithms are the rudimentary building block, the programming is procedural.

To see how the basic building block of a language aids in defining a language consider other types of languages. They use different fundamental elements. Goals are used in logic-oriented programming. Rules are used in rule-oriented programming. Invariant relationships used in constraint-oriented programming [Booch 38].

One interesting variation of procedural programming is event-driven programming. In event-driven programs, procedures are called based on the occurrence of events. These event triggered procedure calls should not be confused with objects' behaviors, since the procedures are not encapsulated with static and dynamic data about the entity the procedure affects.

### Objects and Classes

An object has two definitions. The first definition identifies an object as a member of a group: an object is an instance of some class [Booch 36]. The second definition provides a means of distinguishing one object from another while maintaining a group affiliation:

"An object has state, behavior, and identity; the structure and behavior of similar objects define a class; the term

instance and object are interchangeable" [Booch 77].

Given that group membership is essential to the sense of an object, a language without the ability to define classes (or a template for instance variables) is missing a basic component of object-oriented programming. A language with a frozen set of classes violates the intent of inheritance (the next topic).

## Inheritance

Classes are related to one another by inheritance. Inheritance is endowment that passes from one class to another. For example, a class "irregularPolygon" could endow a class "sail" with the characteristics (such as bounds, vertices, visibility, layer, pattern, and idNumber) and behaviors (such as make, destroy, move, and size). sail in turn could endow "jib" with the characteristics and behaviors from irregularPolygon. In addition, sail could add to or modify jib's endowment. The modified characteristics "luffVertices," "leechVertices," and "footVertices" could be passed along in place of "vertices." In addition sail could modify the behavior "move" to reflect the sail behaviors of closehailed, reaching, running, and luffing.

Without inheritance a program uses abstract data types rather than objects [Booch 36]. Abstract data types encapsulate data and procedures related to the data into a unit, such as Ada's packages.

## Methods and Messages

Methods or behaviors of an object define how an object acts or performs in the system. In the preceding example, the sail had the behaviors closehailed, reaching, running, and luffing. Another object, wind, could have the properties of direction and force.

Messages are requests and returns between objects. For example, a sail object could query a wind object for its current direction. The wind direction could then be used by the sail to determine whether or not to use its luffing behavior.

On the other hand, the wind might change its direction. Changing direction is a behavior of wind. In this case, the wind could request from the sail an execution of the sail's luff checking behavior.

The communications between objects may be implemented (or thought to be implemented) as messages passed from one object to another. Some messages request action, while other messages inquire about data. Requests for data necessitate a return message.

## Meeting Criteria

The concepts of OOP "can be implemented in any language, new or existing" [Howard 14]. A third generation procedural language can be coerced into simulating object-oriented programming. Coercing a third generation procedural language into implementing an object-oriented design does not make the language object-oriented.

Likewise, using some of the nomenclature of object-oriented programming does not make a language object-oriented. Without support for the components of classes, objects, methods, inheritance, and messages a language fails to give guidance in good OOP practices.

Furthermore, developing a product in an object-oriented language using object-oriented techniques does not make the product's language object-oriented. If the product does not facilitate the object-oriented style of programming, then the resulting language is not object-oriented.

For the purpose of this paper, a language will be classified as object-oriented, if the language satisfies the conditions: (1)

objects have state, behaviors and identity and are the basic building block of programs, (2) the programmer can form classes, (3) classes form a hierarchy through inheritance, and (4) objects interact cooperatively.

## EVALUATION OF OPENSRIPT

This section discusses ToolBook and OpenScript properties that relate to the evaluation criteria. The properties are then compared with the criteria of an object-oriented language.

### ToolBook's Building Blocks

ToolBook presents the user with a collection of "tools" that can draw "objects." The drawn objects are part of the building blocks used in constructing an application. Drawn objects consist of graphic objects, buttons, fields, and record fields. Other objects (group, page, background, book, and system) are also used in constructing an application.

Each object has a script for describing behavior. Collectively the properties of a ToolBook object represent the object's state. Among the properties of every ToolBook object is one called "idNumber" used to uniquely identify the object. ToolBook objects have state, behavior, and identity.

OpenScript has a hierarchy among objects for message passing, as shown in Figure 1. The messages are passed from the bottom of the hierarchy to the top of hierarchy until some object recognizes the request [Buchmiller 53]. Thus, the action in ToolBook depends upon objects, and objects are the basic building blocks of programs in OpenScript.

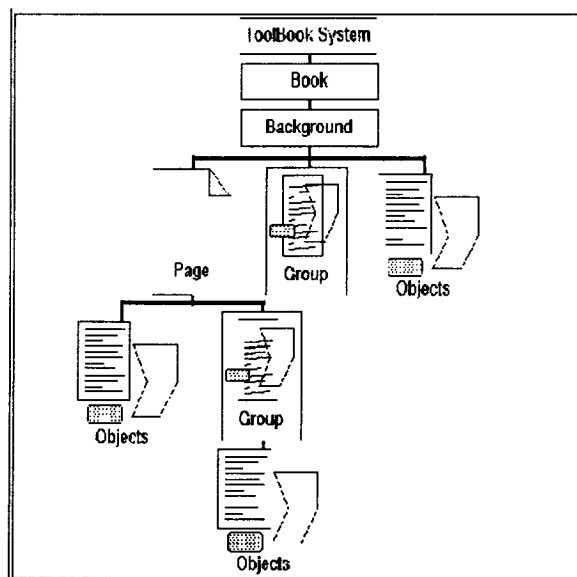


Figure 1: Message Hierarchy

### Creating Objects and Class Hierarchies

"Tools" are used to produce objects. The objects once produced can be copied. The copy has all the characteristics of the original object including the same script.

Through the use of copies, ToolBook has a mechanism for creating objects that inherit properties and behaviors from a source object. These copies have a common set of properties and behaviors. The copied objects might satisfy the concept of "class."

Notice that the user has not declared or created directly a class from which objects arise. Instead, the user has created a collection of objects that have common properties and behaviors. This collection of objects might be interpreted as being a class.

The "tools" that are used to produce objects are more closely related to the concept of class. Unfortunately, a ToolBook user cannot define a new tool; however, HyperCard does support user-defined tools [Brader 15]. If ToolBook allowed deriving a new tool from existing tools, then issues



of class and inheritance would be unambiguous.

## **Inheritance**

If similar copies are accepted as constituting classes, then problems of inheritance arise. Once a copy is made subsequent changes in the original are not reflected in the copy. A copy loses contact with the original.

The copy's loss of contact with the original has an advantage and a disadvantage that are related. The advantage is that a programmer can modify a original without concern for repercussions to copies. By the same token, a programmer cannot correct a defect in the original and have the defect corrected in all of the copies. If the object was replicated many times, the programmer will have the same massive task in fixing an deviant class, as in fixing replicated code.

Replication of objects in ToolBook exists, and a hierarchy of inheritance could be drawn showing the modification of objects and endowed characteristics. Yet, the user-define portions of inheritance is an endowment that can be stripped away. The only properties or behaviors that persist are those endowed by the "tool" that created the original object. All user modifications (and thereby inheritance) can be removed from a ToolBook object. This makes inheritance a matter of programming discipline.

## **Cooperation among Objects**

ToolBook and OpenScript are built on a foundation of objects sending and receiving messages. All actions in the system generate messages. The messages are sent through the hierarchy of objects (shown in Figure 1) until an object recognizes the request and acts upon the message. For demonstrating cooperation

among objects, ToolBook may be unsurpassed.

## **Does OpenScript Meet the Criteria?**

OOP was based on "a visionary new concept of computing. In the vision, non-technical users and even children would interact with computers through the display and manipulation of two dimensional graphical 'objects'" [Howard 18]. If this were the criteria chosen for this paper, ToolBook and OpenScript would qualify as object-oriented development environments.

Booch asserts, "The trend has been a move away from languages that tell the computer what to do (imperative languages) toward languages that describe the key abstractions in the problem domain (declarative languages)" [Booch 26]. If this were the criteria for being an object-oriented language, then ToolBook would qualify.

Non-technical user development capabilities and declarative language style were not the criteria chosen. The criteria chosen were (1) objects have state, behaviors and identity and are the basic building block of programs, (2) the programmer can form classes, (3) classes form a hierarchy through inheritance, and (4) objects interact cooperatively.

Under these criteria ToolBook and OpenScript have two related, major shortcomings: forming classes and supporting inheritance. OpenScript lacks a mechanism for users to define classes. Without user-defined classes, inheritance is an interpretation of ToolBook features rather than an actual feature.

Without an intentional, well-defined mechanism for defining classes and establishing inheritance, an important lesson is lost. Students cannot learn about creating classes. One of the major problems in developing object-oriented programs is the

careful development of classes [Verity and Schwartz 98].

## CHALLENGES OF TEACHING OOP

Programming is a problem-solving activity. From whatever material the programmer is given describing the input and output of the system, the programmer must invent the system. Regardless of what method is used to invent the processing, the programmer has many problems to solve.

When we teach programming, we should be teaching problem-solving. If we do not teach problem-solving, then we are teaching a language (syntax and semantics) not programming. Our students should learn how "to think, to reason, and to build abstractions of the real world" ["Industrial Strength" 16].

Evans describes problem solving, as consisting of several major cognitive functions. The cognitive functions are thinking rapidly of several characteristics of a given object or situations, classifying objects or ideas, perceiving relationships, thinking of alternative outcomes, listing characteristics of a goal, and producing logical solutions [Evans 87-88].

This list of cognitive functions parallels the development of an OOP with amazing accord. The agreement between problem-solving cognitive functions and OOP is not unexpected because "the object model appeals to the workings of human cognition" [Booch 71].

Why then does Terry say, "OOP demands a radically different way of thinking" [Terry 161]? "Unfortunately, most programmers today are formally and informally trained only in the principles of structured design" [Booch 33]. Booch is not belittling structured design. To the contrary, he advises us to remember the valuable lessons learned under structured design.

What he is lamenting is the limited tool kit presented to students. Structured design dictates a solution based on ordering of tasks. Whether the ordering includes concurrent tasks or not the underlying assumption is one of ordered tasks. The single mind-set of ordering tasks makes thinking in object-oriented terms radically different. Thus, in teaching OOP we are adding to the tool kit and having to initiate the transformation from an ordering of tasks mind set to a formalization of the cognitive functions of problem solving.

The transformation from a traditional programming model to an object-oriented way of thinking requires time. "[Michel] Floyd cautions, however, that it takes most people six to eight months to learn to think in object-oriented terms" [Terry 161]. If a tool can help reduce the time required, then the tool would be of benefit in teaching OOP.

Teaching OOP has other competing objectives that complicate the selection of tools used to convey object-oriented concepts. Teachers should "instill in their students the importance of social skills and an appreciation of team effort" [Johnston and Weiss 14]. "Experience in several languages is probably far more important than focusing on just one language" ["Industrial Strength" 16]. Industry is inducing instruction in C++ by "pulling with big salaries for students who know C++," [Lewis 20]. These factors cannot be ignored and will influence which tools are chosen to teach OOP.

## DOES OPENSRIPT MEET CHALLENGE

The tidy objects of text lack a dynamic that illustrates message passing and event-driven programming. Scripting languages provide this dynamic. Pushing a mouse button and having an object respond by way of a student created "to handle but-

tonUp" routine has a tactile and visual appeal.

ToolBook enforces event-driven programming. No cheating allowed. The student is forced to think in terms of objects that bind data and code together. Message sending and receiving are basic to OpenScript programming.

OpenScript and ToolBook are not an object-oriented language by standards defined in terms of text based languages. The declarations are minimal. Inheritance is through objects rather than classes. Classes are intangible.

On the other hand, objects are real. The encapsulation of properties and methods (scripts) is tangible. Code exists in ToolBook only as methods. All code (script) is associated with some object. No code exists without an object to be the actor. Message passing and event-driven programming are fundamental to ToolBook applications.

Could an object-oriented design be implemented in ToolBook, yes. But that is not the measure of an object-oriented language. The measure is does it "guide you into good OOP practices" [Terry 161]. Ambivalence on this count is a strike against a language, and the evidence does not give a clear answer. Therefore, another language would be a better choice for formalizing an object-oriented design into an application.

But ToolBook has a certain visual and tactile appeal that makes it attractive for illustrating OOP concepts, including inheritance.

## CONCLUSION

OpenScript has some of the characteristics of an object-oriented language. However, when the issue of inheritance is considered, OpenScript's properties are less clearly delineated. However, let the text bigot in each of us remember, "There is no

single language that matches all styles and meets all needs" [Rumbaugh et. al. 318].

Howard's advice to "reach into the future and control the OOP revolution" should be heeded. Don't ignore OOP because there is not a perfect language. Pick one (or more) tools for teaching OOP, and concentrate on concepts of design rather than syntax of a language. Until the next programming miracle comes along, we will be looking for the perfect OOP language for teaching OOP. So meet tomorrow, start now exploring OOP and tools for teaching it.

## WORKS CITED

- Booch, Grady. Object-Oriented Design with Applications. Redwood City, California: The Benjamin/Cummings Publishing Company, Inc., 1991.
- Booch, Grady. "Object-Oriented Technology and Industrial-Strength Software Development," Computer Science Syllabus, No 5 (January-February 1993), pp 14-16.
- Brader, Lorinda L. "Tools of the Courseware Trade," Tech Trends, Vol 35, No 5 (Oct 1, 1990), pp 10-17.
- Buchmiller, Carol, Michele DeWilliam, Sharon Dunkel, Annie Pearson, James Purcell, and Nanitti Wright. Using OpenScript: An Introduction and Reference to the OpenScript Language. Bellevue, Washington: Asymetrix Corporation, 1991.
- Evans, James R. "Creativity in MS/OR: Improving Problem Solving through Creative Thinking," INTERFACES, Vol 22, No 2 (March-April 1992), pp 87-91.
- Friedman, Linda Weiser. "From Babbage to Babel and Beyond: A Brief History of Programming Languages," Computer Language, Vol 17, No 17, pp 1-17.
- Howard, Geoffrey S. "Object Oriented Programming Explained," Journal of Systems Management Vol 39, No 7 (July 1988), pp 13-19.
- Johnston, Chris and Jiri Weiss. "Software Engineering with Object-Oriented Programming," Computer Science Syllabus, No 4 (November-December 1992), p 14.
- Lewis, Ted. "Objects Are Closer Than They Appear," Computer Science Product Companion, Vol 1, No 1 (Winter 1992-93), pp 4-6+.
- Milet, Lynn K. and Francis A. Harvey. "An Exploration and Analysis of the Relationships among Object-Oriented Programming, Hypermedia, and HperTalk." Paper presented at the International Meeting of Instructional Systems, November 1989. Available through ERIC: ED 327 153.
- Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. Object-Oriented Modeling and Design. Englewood Cliffs, New Jersey: Prentice Hall, 1991.
- Terry, Chris. "An Object-Oriented Show and Tell," EDN, Vol 36, No 12 (June 6, 1991), pp 161-168.
- Verity, John W. and Evan I. Schwartz. "Software Made Simple," BusinessWeek, No 3233 (September 30, 1991), pp 92-100.
- Wood, Lamont. "Script Languages: The BASIC of the 1990s?" BYTE, Vol 16 (April 1991), pp 244-250.

## University Information Systems Departments Can Successfully Compete in the Commercial IS Training Market

Larry J. Brumbaugh

Applied Computer Science Department  
Illinois State University

### Abstract

This abstract identifies issues related to the role of IS departments in offering professional development courses. Many people attend non-university courses in order to improve their IS expertise, but very few IS departments have an organized program to offer such courses. These courses provide intellectual and financial opportunities for the faculty who teach them.

IS training is provided from various sources including major hardware and software vendors, companies which offer strictly IS training, companies whose IS training component is a major portions of their services but not their only business, small independent companies, often having a brief life span and a very limited geographic market, and individuals. One group conspicuously absent from this list is University and College IS/CS/DP faculty representing their school and department by working as a group rather than as individuals. Most IS/CS/DP departments are not offering training in their area of expertise for industry professionals.

The Applied Computer Science Department at Illinois State University has become heavily involved in PDS training in several distinct settings. The department teaches on-site PDSs for one major client, who seems to have an unending need for high volume IS training. Topics taught most frequently include the IBM mainframe MVS/COBOL environment, systems analysis and design, database, and statistical packages. The department regularly schedules from one to three open enrollment PDSs each month in a nearby mid-size metropolitan area within the geographic region served by the university, which includes several large (thousands of IS employees) IS shops and dozens of medium size (over 100 IS employees) shops. The department also receives requests for individual faculty members to teach specific courses, and faculty subcontract from other IS training vendors.

There are a number of ways in which a department's probability of success in IS PDS training can be substantially increased. First, have one person in charge who aggressively markets the PDS program and runs it on a day to day basis as a business. The person managing the program may not necessarily be a PDS instructor. Second, support people must be available who can help to very quickly develop necessary PDS course materials. Third, school administrators need to realize that presenting PDSs increases rather than lessens their faculty teaching ability and that people who excel in IS training can earn money because they possess a very marketable skill. Finally, realize that not all faculty are candidates for becoming PDS instructors.

**IS THERE A RELATIONSHIP BETWEEN  
FOUR-YEAR UNIVERSITY FACULTY MEMBERS'  
COMPUTER USE AND DEMOGRAPHIC CHARACTERISTICS?**

Dr. Thomas B. Duff & Dr. Patricia A. Merrier  
University of Minnesota, Duluth

**Abstract**

Research studies have been completed to determine the frequency and purposes of computer use by business professionals. However, very little is reported in the literature on the use of computers by one relatively large group of professionals--college and university faculty. Therefore, this study was completed to determine the frequency of and purposes of computer use by such a group.

**Purpose.** This study was conducted to determine what percentage of four-year college/university faculty members use computers in their teaching and/or research activities, whether use of computers is related to certain demographic variables of faculty members, and for which teaching and research activities faculty members most frequently use computers.

**Procedures.** A survey instrument designed to collect data was sent to all 323 individuals classified as faculty members assigned to teach regularly scheduled courses at a regional, comprehensive, four-year university in the Midwest with a total enrollment of about 8,000 students. A total of 230 (71%) usable responses were received. The group of respondents included: 73% male, 71% over 40 years of age, 71% who completed their highest degree in the 1970-1979 period, 65% with between 6 and 25 years of teaching experience, 31% professors, 30% associate professors, and 33% assistant professors. Of the respondents, 29% were from units in Science & Engineering, 24% from Liberal Arts, 18% from Education & Human Services, 11% from Medicine, 10% from Business & Economics, and 7% from Fine Arts.

**Findings.** For the group of respondents, 87% indicated they use computers to support teaching and/or learning activities; all those responding as "users" indicated they used a microcomputer, and 53% of them indicated they used a mainframe or central system in addition to a microcomputer. Based on chi-square analysis, respondents who were older ( $p < .01$ ) and had more total years of teaching experience ( $p < .05$ ) were less likely to use a computer in their teaching activities; and females ( $p < .05$ ), those who were older ( $p < .01$ ), and those who had more total years of teaching experience ( $p < .05$ ) were less likely to use a computer in their research activities. Almost all of those who indicated they used a computer reported they used it to prepare student handouts (98%) and to prepare final narrative research reports (95%). In addition, half or more of those using a computer use it to support managing gradebooks or other class records (50%) as part of teaching and, as part of their research activities, to prepare tables and other summary data (72%), complete data or statistical analysis (59%), and conduct literature searches or retrieval (52%). Although these findings apply specifically to the faculty respondents and their unique environment at the campus involved during the year of the study, they may be generalizable to other faculty members and campuses as well.

# Can Interactive-Audio Televised Instruction Be Effective?: A Review of the Literature<sup>1</sup>

Eli B. Cohen, CDP, CSP, CCP  
Eastern New Mexico University

## Abstract

Interactive-Audio Televised Instruction (ITV) as a delivery system for college instruction has received only limited attention by investigators. Few studies use standard research design and so generalizability of the efficacy of this delivery system is limited. This paper explores those studies and concludes that ITV is a mixed blessing. It is effective in delivering knowledge to motivated students at remote sites, but it also causes unwanted disefficiencies.

## What is ITV?

The term Interactive-Audio Televised Instruction (ITV), as used in this paper, refers to a type of distance education in which students at remote sites can see, hear, and talk to the instructor, but the instructor cannot view the students at the remote sites. ITV is an extension of information technology that allows an instructor's presence to be in more than one location at a time. It differs from videotaped instruction in that it is interactive.

ITV is a form of Distance Education (DE) (Barker et. al., 1989). Some forms of DE have been around for at least seventy years while other forms take advantage of recent developments in Information Technology (Immonem & Rinta-Kanto, 1991). Beginning in the 1920s, students unable to take advantage of traditional classroom education used correspondence courses to learn at remote sites. DE reached new audiences with the advent of television. Since the 1950's, televised educational courses have reached millions of viewers in their homes.

Recent additions to televised course networks include Channel One for elementary schools and Mind Extension/University (National School Board Association, 1992; Crockett, 1993). Both televised courses and prerecorded courses suffer from the same problem as correspondence courses--students cannot interact with their instructor except through correspondence. This limitation led to the next step in DE, ITV.

ITV enables students at remote sites to talk with the televised instructor during class. This ability requires that the instruction be live. Under current technology, the television signal is transported by wires, cables, or microwave to select remote sites that are set up to transmit a voice signal back to the host site. Typically an operator is required to supervise the camera and microphone equipment (Barker, 1992).

The potential of ITV is great. Students in even the remotest location can benefit from specialized instruction typically found only in urban areas (Barker, 1989). Yet for all its potential, current ITV technology is still based in the pas-

---

<sup>1</sup>The author gratefully acknowledges the research assistance of graduate student Laurel Stanley in the assembly of the literature reviewed here and the contribution of the many readers who provided comments and criticisms to earlier drafts of this manuscript.

sive media of television. This paper addresses what experience and studies have uncovered regarding the efficacy of ITV in both the cognitive and non-cognitive domains. The paper also addresses the impact of ITV on teacher planning, delivery, strategies and student interaction. The paper concludes with remarks on how to make ITV more effective.

## **Cognitive Outcomes using ITV Delivery**

The assessments of the effectiveness of ITV are primarily impressionistic, despite its widespread use covering many years and many classrooms. Opinion has it that ITV is as effective as traditional classroom teaching. For example, Barker (1989) reports that available information, although scarce, shows ITV to produce similar cognitive learning as traditional classroom instruction for primary, secondary, and college education.

Barker's summary coincides with studies such as Blanchard's (1989) empirical study that compared ITV to traditional, face-to-face instruction using measures such as attitude of students and faculty, learning, and course grades. Blanchard's study adds credence to the view that those who volunteer to use ITV at remote sites have similar outcomes as those who are assigned to a traditional classroom setting. However the methodology of this and similar studies is flawed because participant sampling was not random.

This interpretation may clarify why the results of Eiserman and Williams (cited in Moore et. al., 1990) study were mixed. Eiserman and Williams could find no study that compared ITV effectiveness across different instructional design, content areas, or types of population. They report that those ITV programs accepted by state accrediting agencies offer little or no empirical evidence to support their efficacy.

## **College Experiences with ITV**

A few studies focused their attention on ITV use in the college setting. For example, Whittington (1987) found that the grades assigned to students at the host and remote sites were not significantly different.

Kabat and Friedel (1990) surveyed students at the host and remote sites of the Eastern Iowa Community College school district and compared them with students taking similar courses on the main campus using traditional instruction. As in the Whittington study, the authors report no statistically significant difference in course grades among the groups.

Of course, course grade is not the best measure of educational effectiveness. Grading on a curve, giving all classes the same proportions of each grade, is common in large classes. Better measures of efficacy are student achievement and student perception of course difficulty. Moore and McLaughlin (1992) used these measures in their study of ITV effectiveness at St. Cloud University. They report ITV learning to be as effective at the remote sites as at the host site.

We need to recognize the limits of these studies. More work is needed to explore the following questions:

- Studies need to consider three learning environments: traditional classroom settings, the host site of ITV, which involves both the professor and students, and remote ITV sites, which involves just students. Is ITV learning at remote sites similar to learning in a traditional setting? This is a different question than comparing ITV learning at remote and host sites.
- How does self-selection of students at remote ITV sites affect efficacy and student motivation? Do students who choose to take a class at a remote site, perhaps to avoid a lengthy commute, have different learning styles than others?

- Is class size at remote sites similar to class size of the traditional classroom?
- How does the host ITV class compare with non-ITV classes at the host site?
- Are teachers selected to teach on ITV randomly or based on criteria related to efficacy?

## ITV Planning and Delivery

The general wisdom regarding planning and delivery of ITV classes is perhaps a truism--the best classes are well-organized, well-planned, and well-managed (Barker and Patrick, 1988). Let us examine what is the same and what is different about excellent teaching on ITV.

### What is the Same?

Many authors argue that effective teaching for ITV uses, in essence, the same techniques that make for effective teaching in traditional settings (Barker, 1992; Johnson & Tully, 1989; Clark, cited in Simonson & Frey, 1989). Barker and Patrick (1988) list these effective-teaching qualities as *use of interaction, use of advance organizers, frequent review, and full understanding of the topic being taught*. These authors believe that effective instruction and curriculum is of greater importance than the delivery systems used to convey this material.

### What is Different?

Other authors believe that the ITV delivery system constrains instruction and so creates demands over and above that of traditional instruction. Gutenko (1991) writes that instructors must compensate for the media to preserve the essence of traditional teaching. Aspects of this compensation include providing vivid examples, interesting stories and careful lesson plans that include plenty of opportunities for student interaction. This compensation is needed in part because the professor cannot receive non-verbal cues from off-site students and also because informal after-class interaction is stifled.

Other differences in teaching are inherent to having the professor on camera (Holmberg, 1989). To obtain optimum results, a technician must ensure that the lighting and sound are up to broadcast standards. Because the communication equipment may (and will) fail, the professor must have contingency plans to accommodate those times when the equipment fails.

ITV requires the professor to adopt new teaching styles. For example, the animated professor who prefers to "effectively use the stage" by walking during traditional instruction must learn to adapt to the camera by sitting or standing in one location. Moore et. al. (1990) add that the traditional use of eye contact, gestures, and facial expression must be modified to be effective on the air. Indeed, professors must adapt to the lack of non-verbal feedback from students (through students' facial expressions) that typically professors use to judge pacing and student comprehension.

Moore and McLaughlin (1992) add that teachers must give the appearance of spontaneity through careful planning, since time limitations do not permit real spontaneity. Henri (1988) refers to this simulation of the tradition spontaneous communication as *guided didactic conversation*, a deliberate, forced interaction.

Lastly, as Moore et. al. (1990) point out, another difference between traditional teaching and ITV involves the extra planning necessary to get materials to and from the remote sites. Since quick feedback is a part of good teaching, this delay encumbers quality instruction.

### Taping of Instruction

While any classroom instruction can be taped, ITV instruction can be taped without further obtrusiveness. This tape can be reviewed by the professor as part of on-going teacher development. It can also be used by students who wish to review the activities of the lesson.



Taping of classes complicates ITV by adding a new dimension that will be explored below-- what are the obligations of the concerned parties regarding this intellectual property.

## **Affective Outcomes using ITV Delivery**

The results cited above deal with the efficacy of ITV in terms of grades and learning as measured on tests. Of course, education is more than a dissemination of knowledge. It also affects the affective domain of how the constituents feel about the ITV delivery system. The discussion below deals with how students feel about ITV and how it affects faculty/student interaction.

### **How Do Students Feel about ITV?**

The thrust of the limited research into how students feel about ITV overwhelmingly shows a negative attitude toward the medium. Barker (1987) found that almost 70% of the students surveyed preferred a traditional setting over ITV. Sixty-five percent of this group report ITV as more difficult.

Barker's findings are in concert with those of Kabat and Friedel (1990). Their survey of college students found that 38% of those at remote sites felt that they learned less than in a traditional classroom. This attitude toward ITV also adversely affects instructor evaluations. Two-thirds of the students rated their instructor awareness as less than good. Simonson, Johnson, and Neuberger (cited in Simonson & Frey, 1989) found similar results among high school students.

The Kabat and Friedel study found that two-thirds of the students would take another ITV class, despite their low esteem for the delivery system. This apparent contradiction can be explained by the factor of convenience.

### **Convenience is Important to Students.**

Moore et. al. (1990) report that their students say that convenience (by not having to travel to the host site) was the most positive benefit of ITV. Despite the students' dissatisfaction with the method of delivery, students want courses that are convenient to their location and schedule.

Dalton (1987) found that despite the importance of convenience, students reported feeling cheated in their ITV experience. It should be noted that Dalton's students experienced an eight-second delay due to limits in the communication system. That delay stifled discussion and undoubtedly contributed to the students' negative attitude toward ITV. ITV systems that use microwave are likely to experience signal fade-out with regularity.

### **ITV Adversely Affects Faculty/Student Interaction.**

Part of the largely negative student view of ITV may be due to limits that it imposes on Faculty/Student interaction. As Dalton noted above, ITV can stifle student question-asking. Indeed, we should not be surprised that interaction suffers when faculty cannot see their students.

In the Moore and McLaughlin (1992) study, students reported that they could not perceive the instructor's facial expressions. The students also criticized their inability to interact with other students or the instructor and the absence of interpersonal relationships. This result is also found in the Alford (1991) study of high school ITV students.

Kabat and Friedel (1990) studied student-teacher interactions on ITV. They discovered that ITV students interacted with the instructor just half as many times as students in a traditional classroom. Even those interactions tended to be short and were reported as "less enjoyable". Whereas 78% of the students in a traditional classroom rated the instructor inter-

actions as adequate, only about half of the remote site students did so.

Alford notes that ITV can lead to social isolation and so may be better suited to the motivated student, the student with high self-confidence. Social isolation was also noted as a problem in ITV by Krajnc (1988).

### **How to Make ITV Effective.**

A review of the literature on ITV would not be complete without a discussion on how to improve the delivery system. The studies cited above point out areas for improvement that can be summarized under technical, pedagogical, and political issues.

#### **Technical Issues**

The criticism of ITV found above is in part due to limits in the one-way visual communication. Dede (1988) suggests that the world can expect to see communications becoming more affordable and more powerful. This prediction is on-course. Improvements in telephone company-based communication called ISDN portends to bring two-way video to not just the remote classroom, but the individual's home. ISDN can also offer students the ability to view tapes of prior classes at home on demand.

Another technical issue involves the skills of the operator. A skilled and dedicated operator is needed to switch cameras and microphones to create a broadcast that captures the essence of the instructor's wishes.

#### **Pedagogical Issues**

As noted above, ITV requires not only excellence in teaching, but places new demands on pedagogy. Television as a delivery system is passive by nature. New, active techniques need to be developed and refined so as to make the

student at the remote sites active learners (Bonwell & Eison, 1991; Nichols, 1990).

One twist on ITV that poses interesting complications, both technical and pedagogical, is combining television with computer technologies. Preliminary studies in this area suggest great potential, but only after significant developments in both areas (Henri, 1988; Hiltz, 1986).

#### **Political Issues**

The politics and reward structure of teaching need to keep pace with technology. Administrators must understand that professors who agree to teach via ITV are taking on a substantially greater load since the demands of the pedagogy are greater. Lessons require greater preparation and skill than the traditional course.

Secondly, colleges must decide whether ITV is to be forced on faculty. Some faculty will find the loss of face-to-face contact with students to rob them of their "psychic salary" and so view ITV as an impediment to their professional life.

The third political issue is that administrators need to understand that students at remote sites are socially isolated and so tend to rate the course and their teacher lower than do students in traditional instruction. If student comments and ratings are given substantial weight in evaluating teaching performance, professors of ITV courses suffer the double whammy of working under adverse conditions with students who resent the delivery system.

Lastly, this television-based technology poses the potential for abuse. Recordings of the ITV instruction are intellectual property that belong to the professor. Colleges that fail to understand this point of law have the potential of stealing from their employees.

## Conclusion

Current implementation of ITV is a mixed blessing. Research, while limited, paints a portrait of an educational delivery system fraught with peril and opportunity. Students, typically those self-selected by its convenience, may learn the academic material to the same extent as non-ITV students, but feel isolated. They lose out on social interaction and do not like ITV as much as traditional education. Teachers using ITV likewise work at a disadvantage. Their task is greater but the rewards are fewer with ITV. We can anticipate an improvement in distance education once ISDN makes two-way interactive video possible and affordable.

## References

- Alford, N. I. (1991, April). "Attitude and communication in the electronic classroom: A closer look at the interactive television system of instruction". Paper presented at the Annual Meeting of the Central States Communication Association. Chicago, IL (ERIC Document Reproduction Service No. ED 339 065)
- Barker, B. O. (1987, January). "An evaluation of interactive satellite distance education". Paper presented at the Annual Meeting of the Southwest Educational Research Association, Dallas, TX (ERIC Document Reproduction Service No. ED 277 534)
- Barker, B. O. (1989). *Distance learning case studies*. Research Report No. 143. Washington, DC: Office of Technological Assessment. (ERIC Document Reproduction Services No. ED 332 661)
- Barker, B. O. (1992) *The distance educators handbook: An administrator's guide for rural and remote schools*. Charleston, WV: ERIC.
- Barker, B. O. & Patrick, K. R. (1988, September). "Teacher effectiveness via interactive satellite: Preliminary findings from observation of three teachers over the TI-TN interactive satellite network." Paper presented at the Annual Conference of the National Rural Education Association, Bismark, ND (ERIC Document Reproduction Service ED 303 295).
- Barker, B. O., Frisbie, A. G., and Patrick, K. R. (1989). "Broadening the definition of distance education in light of new telecommunications technologies". *The American Journal of Distance Education*, 3, 20-29.
- Blanchard, W. (1989) *Telecourse effectiveness: A research-review update*. Prepared for the Washington State Board for Community College Education. (ERIC Document Reproduction Service No. ED 320 554)
- Bonwell, C. C. and Eison, J. A. (1991) *Active learning: creating excitement in the classroom*. ASHE-ERIC Higher Education Report No. 1. Washington, D. C.: The George Washington, School of Education and Human Development.
- Crockett, K. (1993). "Students earn degrees by video." *Amarillo Sunday News-Globe*, Jan. 24, 7D.
- Dalton, D. W. (1987, February). "The effects of individual and team learning on performance during computer-assisted instruction". Paper presented at the Annual Convention of the Assoc. of Educational Communications and Technology Atlanta, GA. (ERIC Document Reproduction Service No. ED 285 531)
- Dede, C. (1989, July). *The evolution of distance learning; Technology-mediated interactive learning*. Washington, DC. Office of Technology Assessment. (ERIC Document Reproduction Service No. ED 325 099)
- Gutenko, G. (1991, May). "Penetrating the glass wall: Creating and retaining the interactive illusion in televised distance education". Paper presented at the Canadian Communications Assoc. Kingston, Ontario, Canada. (ERIC Document Reproduction Service No. ED 337 154)

- Henri, F. (1988). "Distance education and computer-assisted communication." *Prospects*, 18, 85-90.
- Hiltz, S.T. (1986). "The 'virtual classroom': Using computer mediated communication for university teaching." *Journal of Communications*, 36, 95-1-4.
- Holmberg, B. (1989, August). "Distance education with a human face." Paper presented at the 5th Annual Conference of Teaching as a Distance, Madison, WI.
- Immonem, J., Rinta-Kanto, J. (1991) "Features of distance education in Finland." *The American Journal of Distance Education*, 5, 47-52.
- Johnson, L.N., & Tully, S.M. (1989). *Interactive television: Progress and potential*. (Library of Congress catalog card No. 88-64037). Bloomington, Indiana: Phi Delta Kappa Educational Foundation.
- Kabat, E.J., & Friedel, J. (1990). *The development, pilot-testing, and dissemination of a comprehensive evaluation model for assessing the effectiveness of a two-way interactive distance learning system*. Report funded by the First in the Nation Educational Foundation. Davenport, Iowa. (ERIC Document Reproduction Service No. ED 332 690)
- Krajnc, A. (1988) "Social isolation and learning effectiveness in distance education." Paper presented at FernUniversitat. Hagen, West Germany. (ERIC Document Reproduction Service, No. ED 301 667)
- Moore, C.E. & McLaughlin, J.M. (1992). "Interactive two-way television: Extending the classroom through technology." *Technological Horizons in Education Journal*, 19. 74-76.
- Moore, M.G., Thompson, M.N., Quiggley, B.A., Clark, G.C. & Goff, G.G (1990). "The effects of distance learning: A summary of literature". Report for the American Center of the Study of Distance Education. Penn State, PA. (ERIC Document Reproduction Service No. 330 321)
- National School Board Association. (Producer). (1992). *Telecommunications: Bringing the world into the classroom*. [Enteractive teleconference]. Dallas, TX.
- Nichols, J. (1990, March). "R.I.P.--The deadly art of team-teaching with a talking head." Paper presented at the Rural Education Symposium of the American Council on Rural Special Education the National Rural and Small Schools Consortium. Tucson, AZ. (ERIC Document Reproduction Service No. ED 337 331)
- Simonson, M.R. & Frey, D. (Eds.). (1989, February). *Proceedings of selected research paper presentations at the Convention of the Association for Educational Communications and Technology and sponsored by the Research and Theory Division*, Dallas, TX. (ERIC Document Reproduction Service No. ED 308 805)
- Whittington, N. (1987) "Is instructional television educationally effective? A research review." *American Journal of Distance Education*, 1. 47-50.

# **LOCAL AREA NETWORK (LAN): DESIGN AND IMPLEMENTATION FOR A MANAGEMENT AND SKILLS COURSE**

**MARIAN SACKSON, PACE UNIVERSITY**

## **ABSTRACT**

Students pursuing a B.S. in Information Systems need the knowledge of data communications and network management. The increase in the client-server approach for integrating users and the computer, leads to intensifying the significance of networks and network management. Everyday the major newspapers and magazines publish network and data communication articles. In addition, there is a growth in employer demand for entry-level people with network skills.

Data communications theory is easily assimilated into a university curriculum and supported by a plethora of good data communication textbooks. However, a management and network skill's course can be difficult to offer. This paper outlines a rationale and implementation methodology for a hands-on Local Area Network (LAN) course in the undergraduate curriculum of a department of information systems. There are four major problems in the design and logistics for a LAN management and hands-on skill.

1. The technology of LANs is relatively new and very volatile. Many instructors are unfamiliar with the technology and reluctant to offer classes in areas where they do not feel confident.
2. Learning LAN's system commands, studying the hardware, and generally interacting with a network can be very invasive to the network. This problem is further compounded when the network exists for other purposes. In addition, special learning networks are often beyond the budget of a typical university.
3. Designing the curriculum is difficult because in a reasonable class size (24), tasks that require access to the file server need an arrangement of student teams. Each team needs to rotate between watching and experiencing the tasks throughout the course.
4. Often the guardianship and control of academic computers are a part of non-faculty group referred to as Academic Computing. Thus, access to a network is through this organization and may infer managing political interactions.

This paper describes the author's experiences in the design and implementation of a LAN management and network skills hand-on course. The author addresses her solutions to the above problems, the lessons learned from the experience, and makes recommendations for future offerings.

This course is very popular with students. Unfortunately the demand exceeds the ability to offer seats in the course. This is due to two factors. First, students rightly perceive that this course has the potential for improving their job opportunities. Second, the class appeals to students in an information systems curriculum that does not offer many interactive system oriented experiences. Aside from programming classes, students do not have access to hands-on LAN courses in most curriculums in the country. This is, of course, because it is expensive to offer such experiences. There is a definite need for such a course. As educators we are able to use more ingenuity to work within our constraints to offer a well-rounded curriculum.

# MULTIMEDIA AND CAI AUTHORIZING SYSTEMS

John Sigle

Department of Computer Science, Louisiana State University in Shreveport, One University Place,  
Shreveport, LA 71115

## ABSTRACT

In addition to the traditional computer assisted instruction (CAI) programs, a large number of software products exist which can be used to develop computer-based instructional material. This paper surveys some of the available products and discusses their use in preparing instructional materials for use both inside and outside the classroom.

## INTRODUCTION

Computer-based presentation/instruction systems have arrived and are here to stay. The promise of computer assisted instruction (CAI) has been slow to develop. The early drill and practice packages with their mundane textual presentations were generally dull and boring. The unsophisticated interaction and branching was less than inspiring, and consequently it never really caught on. But finally, in the 1990's, the exciting world of multimedia and hypertext brings to CAI the punch it needs to start to realize its promise.

It has been said that people retain about 20% of what they hear; 40% of what they see and hear; and 75% of what they hear, see and do. Multimedia systems provide the tools for producing interactive instructional materials which can capture and hold the attention of today's learners. Even the current generation of graphical presentation systems, which don't generally provide for sophisticated interaction with the user, support the development of attention getting (and holding) presentations.

Whether we like it or not, we are moving into the age of "edutainment", where, as time goes on, the packaging of educational materials will become more and more entertaining. I do not suggest that such materials will replace instructors at any educational level, but rather that these materials will make us more effective and efficient teachers. I also suggest that as educators

we can benefit from knowing about and gaining experience with systems for producing such materials.

Although it is possible to spend \$10,000 or more for a full blown multimedia package, it is possible to get started with little or no expense if you have available a typical personal computer system.

## DEFINITIONS

Since there is a lot of marketing hype associated with this area, let's start with some practical definitions of some of the principal terms.

*Multimedia* is material composed of some, but not necessarily all, of the following elements: full-motion video, still images, animations, computer graphics, text and audio. This material is generally delivered on a computer system.

*Hypertext* refers to the ability to create and browse through complex networks of linked documents. In contrast to ordinary books which are structured as a sequence of pages, hypertext provides a way to navigate through documents in a non-linear, non-sequential manner. In its simplest form a user can choose a "hot word" on the screen and immediately get more information about that "hot word", returning easily to the original screen.

*Hypermedia* is simply multimedia which makes

use of hypertext concepts. This term evolved from the hypertext projects as they discovered multimedia.

*Computer Assisted Instruction (CAI)* refers to any system which is used to instruct by means of computer-based presentation of information and interaction.

*Computer-Based Training (CBT)* is fundamentally the same as CAI. The two terms are generally in vogue in different circles.

## TYPES OF SYSTEMS

The set of products that can be used for the production of computer-based instructional materials runs the gamut, from high priced specialized authoring systems to inexpensive word processors.

This section will discuss three categories of systems for developing and presenting such materials. These categories are not cleanly delineated and there are a few systems that don't fit well in any of these categories. However, these categories provide a useful framework for discussing and understanding the multitude of available systems. The ordering of the categories is from most specialized and powerful (for authoring instructional materials) to least powerful. This ordering generally also corresponds with decreasing prices.

Most of the specific products described run on a standard IBM PC, either under DOS or Windows, with a few running on an Apple Macintosh. A small number of products are available on both platforms. Those that support multimedia generally require extra hardware such as sound cards, CD-ROM drives and video capture and overlay boards.

### CAI/CBT Systems

A number of products are specifically aimed at the complex and laborious task of authoring

courseware, i.e. software that presents instructional courses. Some of these products have evolved over the past decade or more from mainframe CAI products and most have recently incorporated multimedia. They generally have highly structured facilities for lesson planning and development. These typically include the flowcharting of branching possibilities within and among lessons. They usually have fairly sophisticated facilities for judging the student's responses to questions. Some of these systems include a programming language to be used to extend the basic capabilities provided.

Perhaps the most distinguishing characteristic of systems in this category is that they generally have extensive facilities for recording and analyzing the student's performance and for managing and reporting the progress of a student or a class of students. They are also distinguished by their price. They typically carry a list price of between \$2,000 and \$8,000 for the authoring system. Furthermore, in order to distribute the course you author, you must obtain a license to distribute the student (presentation only) version of the product. This may be an additional cost. Fortunately, however, several of these products are offered at substantial educational discounts and with liberal licensing arrangements for the student version.

### Example Systems

Quest Authoring System from Allen Communication is one of the older CAI authoring systems. It runs under DOS on the IBM PC as well as on the Macintosh. It has an extensive set of authoring facilities and incorporates the full range of multimedia features. It includes a Pascal-like language for extension. It comes with extensive course and student management facilities. The educational price is around \$1300 and the student version can be distributed freely without charge.

The Summitt Authoring System from Conceptual Systems, Inc. is similar to the Quest System, but

developed more recently. Its easy to use menu based interface provides access to an impressive collection of authoring features. Pricing for the basic system starts at \$999.

Authorware Professional from Authorware Inc. is a newer product that runs under Windows on the IBM PC as well as on the Macintosh. It uses the flowchart as the paradigm for lesson development. The author develops and expands a flowchart of icon symbols as he/she develops the lesson. Since this product runs under the graphical interface environment of Windows and the Mac, it offers very good graphical presentation facilities. These environments also provide the basic support for the multimedia facilities. The list price of this product is about \$8,000 but it is available at an educational price of just under \$1000 and the student version can be distributed within your educational institution without additional charge.

IconAuthor from AimTech Corp. runs on the IBM PC under Windows. It is similar in many ways to Authorware Professional.

### Hypercard Clones

A number of products are clearly based on the interface paradigm introduced by the Hypercard program on the Macintosh. This paradigm presents sets of screens or windows of information which can be linked together in a hypertext fashion. On these screens or windows appear (mostly) graphical OBJECTS. The standard categories of object include BUTTONS to be clicked on with a mouse to initiate an action, FIELDS of text to be displayed, and PICTURES (photographic and computer generated graphic images). Recently multimedia OBJECTS have been added for manipulating recorded speech and sound, still and live video and animation. These objects can (sometimes) be activated in combination to produce impressive visual and auditory effects.

These products provide for interaction with the

user (student), but that interaction is less structured than with the CAI authoring systems. Consequently, more work is required to develop questions and other dialogue. These systems generally provide no student or course management facilities. If the product supports a programming language, which many of them do, such facilities can potentially be developed by means of that language.

These packages are somewhat cheaper than the CAI packages, ranging from about \$200 to \$800. They generally also allow for free distribution of the authored material.

### Example Systems

Toolbook and Multimedia Toolbook from Asymetrix Corp. are exciting new products which run under Windows and which provide a great deal of flexibility in developing instructional material. They take full advantage of the powerful graphical user interface (GUI) provided by Windows. The multimedia version also takes advantage of the multimedia extensions in Windows 3.1.

Toolbook provides a unique event-driven programming language for extending the capabilities. An interesting feature is that programs (called scripts) can be partially created automatically by simply recording actions performed from the keyboard and mouse.

The basic Toolbook package costs around \$300 and Multimedia Toolbook around \$800, but educational discounts are available. The license to distribute the student version is free.

LinkWay and Linkway Live from IBM run under DOS. Linkway Live is the updated version of Linkway which adds support for full-motion video and provides for more convenient access to multimedia facilities. They are much simpler and less powerful packages than Toolbook. This means that they are easier to learn to use, at least for simple tasks, and are more limited. They





are also slightly cheaper, and the license to distribute the student version is free. They do include a programming language.

### Screen Show Programs and Graphical Presentation Systems

Screen show programs (sometimes called slide show programs) have been around for a decade with Show Partner F/X being an early example. They provide for sequencing the display of a set of screens possibly with special effects on screen transition. More recent products incorporate multimedia. These products have very little support for interaction and therefore no facilities for student record keeping or management. The better ones do, however, make the task of building a presentation quite easy.

Graphical presentation systems have excellent facilities for creating high quality graphics for hardcopy reproduction. Most of them also support computer based presentations as well.

Paint programs, such as Microsoft Paintbrush, CorelDraw, Deluxe Paint and many others, enable users to create computer art. The products in this category, as well as in the categories above, either contain their own paint programs, allow you to import computer art created with other products or both. Libraries of clip-art can be purchased for inclusion in your presentation. However, you need to be careful about compatibility issues such as screen resolution and colors as well as file format of artwork.

Text fonts are another aspect of creating presentations. Again, most of the products have their own facilities for manipulating fonts and many allow you to incorporate fonts from other sources. The Windows environment itself provides powerful font capabilities.

Screen show and graphics presentation programs generally range in price from \$250 to \$600.

### Example Systems

Storyboard Live from IBM runs under DOS and supports animation and other multimedia. It includes its own paint program.

Harvard Graphics now *comes in both a DOS and a Windows version*. It has an outstanding paint program and good facilities for creating a presentation.

Freelance Graphics for Windows from Lotus Corp. is notable for its easy to learn interface and its excellent use of the Windows environment.

Action! from Macromind/Paracomp has excellent support for animation and sound. Action! is available for both Windows and the Macintosh.

### Other Possibilities

Only a sampling of products have been mentioned. Dozens of similar products exist with widely varying price tags, equipment requirements and capabilities.

I would also like to mention the use of word processors/text editors which I refer to as "the poor man's presentation systems". Even if you do not have any of the types of packages described above, you are likely to have a word processor or text editor at your disposal. While these do not generally contain multimedia facilities (Microsoft Word for Windows being an exception) they can be used as simple and crude presentation tools. In fact, my first experience with computer-based presentations was using the Turbo Pascal editor to create class notes and programs for use in a programming course. The notes were presented in class using that editor and a panel display and overhead projector. Incidentally, these notes were also distributed to the students on diskette. One trick which I learned was to simulate animation by paging through the text which contained slightly altered images on each consecutive page.

A Word processor with keyboard macro facilities opens up more possibilities. With a product such as Word Perfect 5.1 which has macro commands which allow for: delaying, turning the screen display off and on during macro processing, pausing until a key is pressed, and nesting of macros, it is possible to do simple but interesting forms of animation. In addition to moving text (or character graphics like lines and boxes) you can change the color of an item, or make it appear or disappear dynamically. With some systems clip art can be included.

### **HOW THESE PACKAGES CAN BE USED**

As mentioned earlier, these packages are useful for developing materials for classroom presentations. The simplest case is to develop and present the lecture electronically using a display panel and overhead projector. You can map it out much as you would on paper. Start with an outline or list of major points to be covered. If hypertext linking is available, use it to move to each item to be discussed. The presentation for each of these main items can be further broken down if appropriate. Principles and definitions can be presented in text form and illustrated by text, charts and drawings, and perhaps animation. If you are a little more ambitious, and the situation warrants it, sound, still images and live video can be used.

Materials can also be developed for use outside the classroom such as in a computer lab. Interaction becomes more important in this situation, and the more specialized products make this task easier. This material could supplement a textbook, or ultimately, replace it.

### **OBSERVATIONS AND RECOMMENDATIONS**

Equipment requirements and complexity, performance (speed), and storage requirements are important considerations. My first suggestion is to not use video unless you really need it. The technology has not quite developed far enough to make video easy and cheap to incorporate.

Sound is somewhat easier and cheaper. Sound cards cost less than \$500 and are easy to use. Keep in mind that a minute's worth of sound will typically require about half a million bytes of storage, so use sound sparingly.

Two primary forms of animation are cycle animation and path animation. With cycle animation, many frames of an object such as a galloping horse are displayed in rapid succession at a fixed location on the screen. Path animation involves moving a single image (usually) along a designated path on the screen. Cycle animation that uses dozens of frames can be very expensive of storage. For example, if the image is bitmapped and uses a quarter of the screen, a dozen frames could consume one million bytes of storage. Path animation can be much less expensive of storage, if a single frame is used. Animation produced by executing a program is generally very economical of storage.

For all graphics images, animated or not, if those images are stored as vector images instead of as bitmaps the storage requirement is lower. This generally correlates with faster performance. Combining multimedia such as sound and animation simultaneously may slow the presentation to an unacceptable rate, so be careful about such combinations.

For all the advantages that the Windows environment provides, it is true that programs run slower under Windows than under DOS. With this environment, especially with the use of multimedia, a fast 386 is needed as a minimum.

On the methodology side, it is important to map out the contents of your material. This is frequently referred to as storyboarding the presentation. This process often turns out to be more work than initially envisioned, but is an important step, especially for larger projects.

Also, for large projects, the team approach works well. Such a team might consist of a subject matter expert, a person knowledgeable in

educational strategies, an artist, and someone with computer expertise. Start by developing a small simple prototype lesson to establish an understanding of the capabilities of the system as well as an understanding of the development process.

Most importantly, get started in computer-based authoring. So much of what you will need to know can best be learned by experience. Authoring can be difficult and time consuming, however, it gets easier with experience.

### REFERENCES

Miller, M.J., "Multimedia", PC Magazine, March 32, 1992.

Miller, R., "To Inform and Convince: Ten Presentation Graphics Programs", PC Magazine, March 17, 1992.

Simone, L., "Presentation Graphics", PC Magazine, May 14, 1991

Multimedia Source Guide, Supplement to the T.H.E. Journal, 1992-93.

Multimedia Solutions, IBM, Sept.-Oct. 1991.  
"The 1993 Multimedia Tool Guide", New Media, 1993.

Computer Magazine, IEEE, Oct. 1991.

Greenfield, E., "Multimedia Authoring: Business Tools Help Education", T.H.E. Journal, Aug. 1993.

T.H.E. Journal, Feb. 1993.

# STRATEGIC FACTORS IN INTERNATIONAL

## INFORMATION SYSTEMS :

### THE GLOBAL PERSPECTIVE

by

Dr. Ronald J. Kizior  
Management Science Department  
Loyola University Chicago

#### ABSTRACT

Many firms are still trying to determine whether The Single European Act of 1987 , which created the European Community (EC) , has affected their operations or not, and possibly what steps should they take in light of this new agreement. There has been a snowball affect regarding the concern for the international dimensions of various phases of business activities. Markets, products, finances are all talked about now-a-days in regard to their global perspective. There seems to be only a splattering of writings about the international concerns relative to information systems ( IS ) and information technology ( IT ). The other functional areas of business, that were mentioned before, seem to have a higher profile or are of greater concern to more corporate executives than IT/IS. It is the author's belief that there are many current business executives , who still might be in the dark, regarding the strategy their company should take, relative to the global perspectives of their information systems activities.

The major emphasis of this suggested course is placed on the concept of enlightenment and awareness of factors relevant to the global perspective of IS and IT . The course that is being suggested here and the topics to be covered are not intended to directly instruct the business executive on how he/she should manage their information system. There are just too many factors that are dynamically linked together to suggest any specific type of action that a company should take relative to its IT and IS strategy on a global perspective.

It is the author's feeling that there are numerous executives who might not be aware of certain factors or may not have thought that certain factors are relevant to their international operations. Two questions that arise are: Is the executive operating a company that is international or global? Is there a difference? These questions and many more will hopefully be answered after the completion of this type of course. During this tutorial a suggested course outline will be covered, and a suggested topical outline for a semester or quarter type course will be discussed and adjusted according to those attending the session. Come prepared to work and adjust this outline to your own specific course and school needs.

# Integrating Database into an Application and Systems Development Sequences for 2nd and 3rd Year IS Undergraduates

**Panel:** Robert Zant, Panel Chair  
Roy J. Daigle  
Herbert E. Longenecker, Jr.  
Robert Stumpf  
Lavette Teague

## Background:

Concepts of data organization and data management are central to development of information systems. Both the DPMA IS'90 and IS'93 curriculum models recommend that database be incorporated into application and systems development sequences.

As opposed to students learning to read text files with Pascal, it makes more sense for beginning IS students to develop Dbase structures and read indexed files. Similarly, COBOL programs can make reference to indexed data files, and to databases with embedded SQL. The primary object is ensure that students master data organization and related data management concepts.

The first assignments may not cover normalization. But, within the context of the usual one year introductory programming sequence, not only can data management, including file processing be introduced early, but normally more advanced concepts including normalization can be blended into the sequence. By the end of a year students could produce validated entry screens, reports, and update programs within a database environment.

Then, the traditional systems sequence could be modified. The new issues would become, "how to determine the database", and "how to manage a database project". By the end of the sequence, students teams would have practiced project management by implementing enterprise level applications. These lab and lecture experiences would only occur with significant faculty guidance.

## Questions:

1. Is it really important to integrate data organization and database concepts in the traditional "COBOL course"? or the "Systems course"?
2. What are the significant and important concepts of data organization and data management that should be covered?
3. To what extent is there any example of successful sequences carrying out the objectives of IS'90 and/or IS'93.
4. What will it take to achieve the proposed curriculum objectives?
5. How can faculty best work together to meet the demands of the new curriculum?

**TEACHING ETHICS IN IS COURSES:  
Everything You Always Wanted To Know But Were Afraid To Ask**

**Ernest A. Kallman & John P. Grillo  
Bentley College  
Waltham, MA 02154-4705**

**Abstract**

We, the presenters of this workshop, would like to start with an admission, which we will assume is also true of the majority of the audience, as teachers in a computing discipline: We're a lot of things...but we're not trained ethics teachers.

Instructors in IS and CS programs come from varied backgrounds including math, engineering, business and the liberal arts. This is still true even with the number of CS and MIS degrees now being granted. But few teachers in these technical disciplines, if any, have backgrounds in philosophy or ethics. We naturally feel uncomfortable tackling computer ethics in the classroom.

**THE PURPOSE OF THE WORKSHOP**

The workshop is for teachers, students, information systems professionals, and computer users. It is meant for those who (1) want to know more about ethics, (2) want to gain some experience dealing with ethical dilemmas, and most of all, (3) may be called upon to teach computer ethics.

Furthermore, the workshop is constructed with the inexperienced computer ethics teacher in mind. A computer ethics teacher does not need to become a philosopher in order to deal effectively with the subject matter. Sufficient background will be provided in ethical decision making and its underlying principles. The connection between information technology and ethics is also explained. Real world cases are used to illustrate the steps for analyzing a specific situation and making defensible ethical decisions.

**WORKSHOP OVERVIEW**

Among the topics to be covered are:

- Ethical decision making
- What makes computer ethics different
- Making ethics happen in the classroom
- Case analysis techniques
- Choosing cases
- Using cases: requirements for teacher and students
- Managing the discussion
- Managing the classroom
- Where to find the time
- How to evaluate student learning

# EVALUATION OF CURRENT DATABASE TEACHING AND ITS FUTURE DIRECTION

## A Panel Discussion at ISECON 1993

Moderator: Joobin Choobineh, Texas A&M University

- Panelists:
1. Hoffer, Jeffery A. Indiana University, Bloomington
  2. Kroenke, David M. Wall Data Inc., Seattle
  3. McFadden, Fred R. University of Colorado, Colorado Springs
  4. Ram, Sudha University of Arizona, Tucson

### ABSTRACT

This panel will discuss some important issues related to modern teaching of database courses within the business curriculum. These issues include the management of students field projects, the emerging division of database products and technologies, inclusion of object-oriented databases in the curriculum, and the importance of teaching alternative approaches to database design.

### Introduction

The first database courses for business majors appeared in the 1970's. The course focused on understanding the hardware and technical aspects of databases. The primary topics in the course were external memory structure, hierarchical model, network model, and to some extent relational model. During the 1980's, with the rapid proliferation of the relational databases, the course was focused on the relational model. In addition, some instructors put some emphasis on data modeling aspects of databases.

As educators of the database topics, it is crucial for us to understand where we are and where we are heading. The purpose of this panel discussion is two-folds. First, to identify what is the current practice of database teaching at business schools, and second what should its focus be during the rest of the 1990's.

In what follows a brief synopsis of each topic, in each of the panelist's own words, is provided. Following the panelists presentations,

the floor will be open for discussions and questions from the audience. Given the availability of equipment, this panel discussion will be video-taped in its entirety.

### The Management of Field Projects. (Professor Jeffery A. Hoffer)

Many schools require student teams to develop an operating database application for a real organization (student club, university unit, small business, human services organization, or division of a large company) as part of the database course. The purpose of such a project is typically to integrate various course topics and skills, and to gain valuable experience working in teams and in real organizations.

Various issues arise in how to manage students through this process. Professor Hoffer will raise some of these issues and outline how the Indiana University curriculum deals with this course activities. Some of the issues are: student and instructor liability, coordinating with field

projects in other courses, how to get students started early on the project, staged deliverables to keep students on schedule, project scoping, and role of instructor. Hopefully this presentation will lead to observations from other programs that could lead to guidelines for us to follow to improve the management of such course projects.

### **The Emerging Division of Database Products and Technology**

(Mr. David M. Kroenke)

Issues raised by the emerging division of database products (and technology) can be placed into two major groups: front-end application development products such as Microsoft Access and Borland's Paradox for Windows and back-end engines like SQL Server, INFORMIX, ORACLE, even DB/2 or OS/400, and all the products accessible from Open Database Connectivity or the Integrated Database Application Program Interface. This division raises several questions: Do both of these topics belong in one course (perhaps the front-end products and technology should be considered in a systems development course?) What should be the relative amount of coverage of each? Then, given the resources available to most database teachers, what topics in these areas can, realistically, be taught?

### **Responding to Object-Oriented Databases**

(Professor Fred R. McFadden)

Object-oriented databases are increasingly being introduced to meet the requirements of complex, large-scale, data-intensive applications. Emergence of these systems raises several issues for educational institutions: how can we best introduce object-oriented concepts and methods in our database courses? What are the essential topics, and how much time should be allocated to them? Should there be a separate course that includes object-oriented programming? How can we provide students hands-on experience, given limited time and budget? By discussing these issues, perhaps we can develop some guidelines or suggested approaches in this important area.

### **Use of Tools, Alternative Approaches to Database Design, and Team Project Agreement** (Professor Sudha Ram)

This discussion will focus on issues dealing with the type of DBMS to be used in the course, choice of a database design tool, and the scope of team projects in the course. Typical questions that need to be addressed on each of these issues include:

- Choice of appropriate DBMS: Which DBMS should be the focus of the course? What kind of implementation experience would be useful to students in order to make them attractive to recruiters? Should the course focus on a microcomputer based system or a mainframe based system? Or does it really matter?
- Database design tool: Conceptual modeling is considered a very important facet of the database design life cycle. To support this stage, what kind of software tool would be useful in the course?
- Relational Database design using the synthesis approach: Most database books and courses cover the decomposition (normalization) approach to relational design. An alternative is the synthesis (Bernstein's) approach. How and why this alternative approach should be included?
- Team Projects: Typically, a class is divided into a number of teams who work with a local organization on a project. What should be the scope of the project and what sort of agreement should be made between the team and the local organization such that both could benefit from the project?



## CURRICULUM IMPLEMENTATION - THE ACM TWO-YEAR COLLEGE EXPERIENCE

### PANELISTS

Karl Klee, Jamestown Community College, NY  
Chair, ACM Two-Year College Committee

Michael Wolf, El Paso Community College  
Chair, ACM CSS Report Group

Suzanne Gill, Mount Royal College, Canada  
Chair, ACM CIP Report Group

Mary Jo Haught, MJ Services, VA  
Member, ACM CIP Report Group

Rod Southworth, Laramie County Community College, WY  
Member, ACM CIP Report Group

Tony Mann, Sinclair Community College, OH  
Reactor

### ABSTRACT

The release of the report of the Association for Computing Machinery's (ACM) Two-Year College Education Committee in the fall of 1993 was the result of several years of activity by the Task Force of about 35 people. The report, completed with the involvement and cooperation of representatives from the Computer Society of the IEEE-CS and the EDSIG group of DPMA, covered four areas of curricular interest as well as the area of service courses for other disciplines.

The report provides curriculum guidelines in these four areas: computing sciences, computer engineering technology, computer support services, and computing in information processing. The fifth report concerns computing for other disciplines. An executive summary is also provided.

The two reports that are of most interest to the ISECON audience are: "Computing in Information Processing" and "Computer Support Services." This panel will address issues of implementation of these two reports. Reaction to the recommendations in the five reports will also be provided.



# TOTAL QUALITY MANAGEMENT IN INFORMATION SYSTEMS: ISSUES AND PROGNOSIS

## Panelists

Don Beaver, Manager, Information Systems  
AlliedSignal Airline Services

John Boughter, Manager, IS Operations  
Salt River Project

Richard Discenza, Professor of Production Management and Information Systems  
University of Colorado at Colorado Springs

Neil Jacobs, Associate Professor of Computer Information Systems and Management  
Northern Arizona University

Bob Kenney, Manager of Sector Software Engineering Process Group  
Motorola Semiconductor Products Sector

## ABSTRACT

Practicing IS managers actively involved with TQM provide their insights on three important issues related to the implementation of TQM in IS. Academic panelists provide a framework to relate these issues to the broad domain of issues facing IS organizations implementing TQM. Each panelist offers their prognosis for TQM in IS.

To provide an overall context for the discussion, Neil Jacobs presents a framework for examining Total Quality Management (TQM) implementation in information systems (IS) organizations. The framework (see Exhibit 1) suggests that the relationship of the attributes of TQM to key information systems processes defines the scope of issues involving implementation of TQM in information systems.

Don Beaver addresses the issue of determining the right level of customer satisfaction to seek and poses these questions: What is information systems' responsibility to its customers? Should IS seek the same level of customer satisfaction for internal and external custom-

ers? Is full satisfaction of internal customers a desirable goal? How can an IS manager strike an appropriate balance between the conflicting goals of full customer satisfaction and prudent control of IS expenditures?

Don contends that different levels of satisfaction are appropriate for external and internal IS customers. He describes a way for determining the appropriate level, that is, getting executive management to make the decision on the balance question. He discusses what an IS manager can do to pose the question and elicit an executive management decision.

John Boughter focuses on the issue of implementing TQM in IS without the umbrella of an organization wide program, a challenge he faces at Salt River Project.

Often, TQM is implemented as a corporate-wide initiative with all business units participating. What happens when a business unit such as IS initiates TQM without a corporate sponsored TQM program? Can IS successfully implement TQM in such an environment? John Boughter discusses some of the activities, processes, and techniques being employed to foster a TQM approach in such a setting. He relates the obstacles and challenges encountered as he used TQM to improve the delivery of IS products and services in the absence of a corporate-wide TQM program. He offers suggestions for other IS managers following similar paths.

Bob Kenney discusses a Motorola approach to the issue of balancing top-down support with bottoms-up ownership of change.

Balancing top-down management support for change with bottoms-up ownership of that change is critical to the success of a TQM implementation. The proper and timely use of organizational change management techniques improves the probability of achieving the company's process improvement objectives. Once a continuous process improvement discipline is institutionalized, organizational change management can become a standard element in introducing new information technology.

Motorola has adopted the Software Engineering Institute (SEI: Carnegie-Mellon University) model for software process improvement. Motorola's Semiconductor Products Sector Information Systems group uses an internal "SEI Challenge Program" to improve the bottoms-up ownership of software process improvement through implementation of the

SEI model. Defined metrics within the program have shown steady improvement in software process maturity. Paired with management support for the SEI model, the "SEI Challenge Program" and its follow-on activities are being highlighted in Motorola's Total Customer Satisfaction Showcase.

Richard Discenza uses the Framework for Assessing TQM Implementation in IS (Exhibit 1) to map the issues discussed within the broader context of the many other issues involved in the implementation of TQM in IS. Further, he identifies some of the most troublesome issues not addressed by the panel.

Given this broader perspective on the issues of TQM implementation within IS, each panelist provides their prognosis on the future of TQM in IS.

The panel encourages session attendees to join in the discussion.

**Exhibit 1**  
**Framework for Assessing TQM Implementation in Information Systems**

	<b>Information Systems Process</b>						
<b>TQM Attribute</b>	<b>System Development</b>	<b>System Maintenance</b>	<b>System Integration</b>	<b>Data Center Operations</b>	<b>End-User Support, e.g, computing, training, and network administration</b>	<b>Establishing and maintaining telecommunications infrastructure</b>	<b>Data Base Development and Administration</b>
<b>Customer Focus</b>							
<b>Fact-based decision making</b>							
<b>Team-based consensus</b>							
<b>Employees empowered to make decisions</b>							
<b>Facilitator vs. leader role for management</b>							

## ACCREDITATION ISSUES FOR PROGRAMS IN INFORMATION SYSTEMS

### PANELISTS

Joyce Currie Little, Dept. of Computer & Information Sciences  
Towson State University, MD

David Feinstein  
Dept. of Computer & Information Sciences  
University of South Alabama

Tom Kirk  
 DeVry Institute of Technology

John Gorgone, Dept. of Information Systems  
Bentley College

John Impagliazzo, Dept. of Computer Science  
Hofstra University

The creation of the Computing Sciences Accreditation Board [CSAB] in 1984 by the Association for Computing Machinery [ACM] and the Computer Society of the Institute for Electronic and Electrical Engineers [IEEE-CS] was in reaction to concerns for the quality of programs in the computer and information sciences discipline. One accrediting commission has been created by CSAB to date - the Computer Science Accreditation Commission [CSAC], which now accredits programs in computer science. The only other agency authorized to accredit programs in the computing field is the Accreditation Board for Engineering and Technology [ABET], which accredits programs in computing offered under Engineering.

At this time there is no accreditation process available for programs in information systems (IS), computer information systems (CIS), or information sciences (ISC). There has been some concern in the past about the need for such an accrediting process. There is some question about whether DPMA or other associations should work toward the creation of such a process.

This panel will provide perspectives on the accreditation of IS/CIS/ISC programs. Information will be provided on existing accrediting bodies and on what has been done in the past to promote the creation of such accreditation. Panelists will give their view of the importance of accreditation, the issues involved in the implementation of an accrediting process for IS/CIS/ISC programs, and their personal recommendations to the information systems education community and DPMA.

# Framing Object Orientation for the IS Curriculum

Workshop

Les Waguespack, Ph.D.

Computer Information Systems Department, Bentley College

175 Forest Street, Waltham, MA 02154-4705

(617) 891-2908

EMAIL: LWAGUESPACK@BENTLEY.edu

## Abstract

The objective of this workshop is to provide IS faculty and program planners an understanding of object orientation as it relates to the needs of the IS community and the preparation of future IS professionals. This workshop is designed to frame the many facets of object orientation, to provide the participant with a comprehensive overview of the object orientation phenomenon and to offer direction on picking and choosing facets for integration into IS curricula. The workshop introduces the foundation concepts and explores how the object orientation model of systems impacts analysis, design, programming, and data management. The workshop does not assume any prior knowledge of object orientation and provides the participant with detailed presentation materials and a bibliography for further individual exploration. Participants are encouraged to share their views and insights.

## Workshop Outline :

### 0. Overview

### I. Evolution of Object Orientation

#### A. Programming language roots

- Simula, Smalltalk, C++, Object Pascal(s), CLOS

#### B. Modeling roots

1. Abstraction
  - a. procedural
  - b. data
2. Modularization
3. Encapsulation

### C. Forms of Object Orientation in Practice

1. Object Oriented Analysis
2. Object Oriented Design
3. Object Oriented Programming
4. Object Oriented Data Management

### D. IS vs. CS Vision of Object Orientation

1. Science
2. Industrial
3. Academic

## II. Essence and Vocabulary of Object Orientation

### A. Terminology with examples

1. Object
2. Class (object type)
3. Inheritance
4. Messaging
5. Polymorphism
6. Encapsulation
7. Abstract Classes
8. Class Library

### B. Varied Language Interpretations of Object Orientation

- Smalltalk, C++, Object Pascal(s)

## III. Object Orientation as System Model

### A. Modeling systems as objects & classes

### B. Object Oriented Analysis

### C. Object Oriented Design

### D. Object Oriented Database

- E. Object Oriented System Architecture
  - communications, client-server, distributed systems

#### IV. Contrasting the Object Oriented Vision with Traditional Models

- A. Process driven modeling
- B. Data driven modeling
- C. Behavior driven modeling
- D. Assessing the Abstract distance between models

#### V. Integrating Object Orientation into the IS Curriculum

- A. Alternative Strategies for Integration
  - 1. Horizontal Integration (across existing courses)
    - a. benefits (to students, curriculum, program, industry)
    - b. costs (for faculty, for laboratories, for changing curriculum)
    - c. risks (for faculty, for program, for market image)
  - 2. Vertical Integration (within specialized courses)
    - a. benefits (to students, curriculum, program, industry)
    - b. costs (for faculty, for laboratories, for changing curriculum)
    - c. risks (for faculty, for program, for market image)
  - 3. Full Integration (changing the underlying system model in all courses)
    - a. benefits (to students, curriculum, program, industry)
    - b. costs (for faculty, for laboratories, for changing curriculum)
    - c. risks (for faculty, for program, for market image)

- B. Piloting Vertical Integration
  - 1. Pilot course description
  - 2. Pilot course syllabus
  - 3. Pilot course laboratory facilities

#### VI. The Future of Object Orientation in IS

- A. C.A.S.E.
- B. Embedded Object Orientation Technology
- C. Supplanting of Traditional Models
- D. Importance to IS professionals and students



**Achieving Quality:  
New Roles for the IS Professional  
in Supporting End-user Computing and Business Process  
Redesign**

Ed Altman, Wall Data, Inc.  
Jon Clark, Colorado State University  
David Kroenke, Wall Data, Inc., Panel Moderator  
Herbert Longenecker, Jr., University of South Alabama  
Jim Mothershead, Wall Data, Inc.

**Abstract**

This panel will discuss information systems quality from two different perspectives. First, each panel member will describe an important aspect of information systems quality. Dr. Longenecker will discuss the results of a recent survey he did to assess end users' and information systems professionals' attitudes toward systems quality. Then, Dr. Clark will describe the inter-relationship of quality and business process redesign. Mr. Altman will discuss a case study in which end-users were actively involved in quality assurance and finally Mr. Mothershead will describe contemporary product quality assurance processes as used in an object-oriented development environment. A system that performs automated systems testing for Windows-based applications will be demonstrated.

Given this background, the panel members will then each state an opinion with regard to actions that the educational community should take to improve information systems quality. These opinions will address curricula standards, course content, course projects, and related matters. The agenda will allow for thirty minutes of questions, answers, and discussion. We hope that the opinions stated by the panel members will invoke a lively dialog with the audience.