Journal of Information Systems Education

Fall 1994

Applying a Framework for Software Development Methods in an Information Systems Curriculum

ABSTRACT: Software developers are constantly exploring new software development methods that are timely and cost effective, and also foster productivity and better quality systems. "A Software Development Method (SDM) is a system of technical procedures and notational conventions for the organized construction of software" [Karam and Casselman p. 34] Karam and Casselman [1] provide a framework for evaluating SDMs within an organization. There are 14 technical properties, 5 usage properties and 2 managerial properties. Is this framework viable for cataloging SDMs in an academic setting, thereby providing guidelines for selecting an appropriate software development methodology for an Information Systems curriculum? The main purpose of this paper is to demonstrate the use of a cataloging framework for SDMs within a business Information Systems curriculum. Information Engineering (IE) and Software Engineering (SE) are SDMs used to illustrate the usage of framework. The framework can be expanded to include additional SDMs. The Washington Post (February 24, 1993) forecasts a 78% increase in the demand for system analysts within the next 10 years. It is important for Information Systems' graduates to know state of the art techniques, and therefore, courses must constantly be updated to meet this demand.

Mary J. Granger
Assistant Professor
Management Science Department
The George Washington University
Washington, D.C. 20052
granger@gwuvm.gwu.edu

Ross A. Malaga
Information Engineer
Ogden Government Service
3211 Jermantown Road
Fairfax, Virginia 22030
70252,3367@compuserve.com

KEYWORDS: *Information Systems Curriculum, Software Development Methodology, Framework, Cataloging Framework*

INTRODUCTION

Because current business Information Systems (IS) are larger, deal with unstructured tasks and process more data, application software complexity has increased. Software developers are constantly exploring new software development methods that are timely and cost effective, and also foster productivity and better quality systems. "A Software Development Method (SDM) is a system of technical procedures and notational conventions for the organized construction of software" [1 p. 34]. "The results tend to indicate that in spite of two decades of experience in developing software, the question of the best methodology is still largely unsettled" [2 p. 38]. As business organizations re-evaluate software development methodologies, so must business schools re-evaluate their IS curriculum. Faculty preparing IS graduates for careers in software development are confronted with a variety of SDMs; they can use some assistance with the selection process. The changing requirements and methods for system development have implications throughout the IS curriculum, therefore, the choice of a SDM is not a trivial decision. The Washington Post (February 24, 1993) forecasts a 78% increase in the demand for system analysts within the next 10 years. It is important for IS graduates to know state of the art techniques and, therefore, the IS curriculum needs to be constantly updated to meet this demand.

The main purpose of this paper is to demonstrate the use of a cataloging framework for SDMs within a business IS curriculum. Academic environments, similar to business environments, are unique; one SDM is not appropriate in every environment. This classification provides some guidelines, not recommendations. The operative framework is one developed by Karam and Casselman [1]; it provides an unbiased method for evaluating SDMs. Information Engineering and Software Engineering (structured methodologies) are SDMs used to illustrate the functionality of the framework. Information Engineering (IE) methodologies focus on the data and the strategic aspects of the entire business organization. Although developed during the late 1970s and introduced in 1982 by Finkelstein, IE has only recently gained popularity. Introduced in the mid-1970s, Software Engineering (SE) methodologies stress the organization's procedures and processes for a single application or project, and are the most widely accepted SDM.

Karam and Casselman [1] provide a framework for evaluating SDMs within an organization. In order to provide guidelines for selecting an appropriate software development methodology, this framework is applied in an academic setting, specifically the undergraduate business IS field.

BUSINESS INFORMATION SYSTEMS CURRICULUM

Emerging methodologies and technologies drive business IS curriculum and research. Business organizations want to employ IS graduates with problem solving skills, an understanding of the business organization and a knowledge of appropriate software development methodologies. The constant changes in methodologies and technologies create pressures when selecting IS curriculum content.

There may be one or more project oriented courses in an IS curriculum. Undergraduate student projects, even if they are 'real-world' systems for a business organization, are often constrained by semester/quarter time and available computer resources. Even with a strongly enforced prerequisite structure, students enrolled in project oriented courses may have widely different knowledge bases. Because Computer-Aided Software Engineering (CASE) tools are often provided to the university by a vendor at either no cost or at a substantial reduction, SDM selection is predetermined. Additionally, the SDM taught in the project

oriented course influences the content of other IS courses. Prerequisite courses, such as programming, prepare the student for the SDM course. Database courses or other subsequent courses build on the knowledge gained from the SDM course. Due to repercussions throughout the IS curriculum and demands from prospective employers, selection of the software methodology demands careful consideration. Ultimately, IS faculty are trying to determine the best curriculum for preparing their graduates for careers in IS.

"A CATALOGING FRAMEWORK FOR SOFTWARE DEVELOPMENT METHODS"

Software development methodologies often deliver less than they promise; a technique is needed for determining the strong and weak components of a SDM. However, there are few frameworks [3][4][1][5] available for SDM evaluation within a business organization; there are no frameworks specifically targeting an academic setting. The Karam and Casselman [1] framework incorporates not just the technical properties of SDMs, but also the usage and managerial properties. It appears to be more comprehensive than other frameworks. The framework is fairly easy to understand and can be readily applied to groups of SDMs, or to a single SDM. It appears to be a viable framework for cataloging SDMs for an IS curriculum.

Karam and Casselman [1] developed a general framework for cataloging software development methods for enabling practitioners to select a methodology or collection of methodologies. There are 21 SDM properties, grouped into three categories: technical, managerial and usage. As the category names imply, technical properties evaluate the SDM's clarification (notation and procedures) of technical problems, managerial properties concentrate on managerial considerations (staffing, cost, planning) and usage properties deal with practical issues (training, tools). These properties are not based on tool support, they determine which SDM is appropriate for the organization or the application.

Software Engineering and Information Engineering are the two SDMs used throughout this paper to demonstrate use of the 21 framework properties. However, since project oriented courses are simulations of 'real world' projects, some of the properties are inappropriate or difficult to apply to an academic setting. Due to the impact on the IS curriculum and IS graduates, it is felt that a systematic approach to selecting a SDM is as critical in an academic setting as in the business setting.

SOFTWARE ENGINEERING

Software Engineering applies engineering principles to software development, organizing software projects as engineering projects. Initially, the formal methodologies of SE were applied to scientific or technical software such as compilers and operating systems; currently, they are also being used to develop information systems and business application software [6][7]. Constantine and Yourdon, Gane and Sarson, Warnier-Orr, and Jackson methodologies are process driven [8] techniques. They emphasize the functional requirements of the system being developed, concentrating on activities required to complete the task.

The three formal methodologies of SE that have a major impact on system development and productivity are structured programming [9], structured design [10] and structured analysis [11]. These formal methodologies emphasize, at different levels, the processes and procedures of the organization.

Structured programming is the writing of a computer program in a standardized manner to decrease debugging and testing problems, increase documentation and readability, and facilitate maintenance. The emphasis is on writing clear, concise, more readable and less error-prone code. Yourdon [12 p. 8] defines structured design as "the art of designing the components of a system and the interrelationship between those components in the best possible way." The major graphical design tools for structured design are structure charts and data flow diagrams. Structure charts emphasize the procedural aspects of the system. They depict the system modules and the interactions between them; a hierarchical order controls the graphic representation of the system. Data flow diagrams represent the data that flow between the processes of the system and the way those processes transform the data. Initially, structured analysis defined all the system requirements in narrative form [12 p. 123]. Currently, requirement specifications are graphical representations used to interface with the user/client.

INFORMATION ENGINEERING

Information Engineering is a data driven software development method. The IE methodology focuses on the way the organization does and will do business. Driven by changing business environments, this methodology incorporates the organization's long-term goals and objectives into an Information System which, in turn, mirrors the strategic plan.

Initially, a strategic plan is created to define the goals and strategies of the organization. IE models the strategic plan, identifying vital organizational data and functions. After modeling the strategic plan, the business areas of the organization are targeted for design. Entities and their relationships defined for the organization are subsequently used in the analysis of the organization's business areas. The strategic data model drives system development at all levels.

IE combines entity relationship (E-R) diagrams [13] and relational theory [14]. Initially, business data models are simplified into third, fourth, and fifth business normal form [15]. Then business processes are identified and derived from the normalized E-R diagrams.

APPLYING THE FRAMEWORK FOR BUSINESS INFORMATION SYSTEMS CURRICULUM

TECHNICAL PROPERTIES

There are 14 technical properties. Many of these properties are based on the assumption that the methodologies have the standard system development life cycle (requirements analysis, requirements specification, architectural design, detailed design, implementation and evolution) and evaluate the SDMs on per-phase levels. Since SE and IE have different software development life cycles (SDLC), discussion of each of the technical properties relates to the whole methodology, not to individual life cycle phases.

Philosophy

The main focus of the IE methodology is the strategic, long-term planning for the business organization emphasizing current and future requirements and priorities. SE methodologies concentrate on an organization's short-term, immediate projects and resources. Both methodologies are highly structural, concentrating on interacting components, but at different organizational levels. IE tends to take into account effects of external events (behavioral); SE is more functional, concentrating on data transforms [1]

Life cycle

Software Engineering and Information

Table 1. SYSTEM DEVELOPMENT LIFE CYCLE

LIFE CYCLE STAGES	SE	IE
Logical Design Phase		
Project Inception	X	X
Requirements Gathering	X	
Systems Analysis	X	
System Design	X	
Specification Review	X	
Business Modeling		X
Process Modeling		X
Enterprise Modeling		X
Logical Design Review		X
Physical Design Phase		
Database Design	X	X
Programming	X	X
System Testing	X	X
Production Turnover	X	X
Project Production	X	X

Engineering methodologies both develop software systems in an iterative, top-down manner. Text books offer different versions of software development life cycles for each of the SDMs. Table 1 lists one version of the software development life cycle [16] for each of the SDMs.

Major work products and notations

There are three levels of management in every organization: strategic, tactical and operations. Both SE [17] and IE [18] support these levels. Organizations and text books observe various work products or different versions of work products for each of the SDMs. Table 2 lists some of the more popular work products and their notations. Versions of the SDLC for either SDM may not advocate all work products listed. In an academic setting, the number and depth of coverage of work products is dictated by the length of time in the semester/quarter. Often, meaningful projects are developed using a subset of the available work products.

Problem domain analysis and understanding

The SE methodology analyzes specific application areas or domains of the organization. Depending upon the type of software being developed, there are specific recommended techniques (data flow diagrams, hierarchical diagrams, Warnier representations) [19]. Before designing any specific application areas, the IE methodology requires a global definition of the organization's data requirements (data models) [20]. This global definition is a superstructure for detailed work and assists in defining the business organization's strategy.

Procedure/Guidelines, criteria, measures

The procedure property and the guidelines, criteria and measures property prescribe per-phase narrative and per-phase rating. After identification of a procedure for each life cycle phase, the consistency of the associated procedure is rated. Each methodology has a definite set of recommended per-phase procedures that should produce dependable outputs from each phase. Since practitioners may not be consistent when implementing these procedures, it is doubtful whether or not students, when applying the same procedures, will obtain similar results. Song and Osterweil [21] are laying the groundwork for this level of evaluation. There is a lack of standards and metrics in IE [8]. Much work has been done on SE measurement and metrics [22][23][24][25][26]. In an academic setting, these measures are often discussed, but rarely have an impact during the implementation or rating of the student projects.

Verification

Both SDMs advocate verification of SDLC stages. Within an academic setting, verification of work products is accomplished by manual inspections. Peer reviews of work products (data flows, data models) are simulated; the feedback often generates modifications to the work in progress. These inspections and reviews are used to find logical flaws. If the SDMs are automated, syntactical errors are easily identified, but formal proofs of the correctness of the work product, while discussed, are rarely applied to student projects.

Degree of formality

Formality is [1p. 38] dependent upon the SDM notations, the relationships between phases and verification. There have been attempts [27][28] to apply precise, mathematical definitions to designs generated by structured design methodology. These proofs are rarely, if ever, used in either a business setting or an academic setting. Advocates of IE apply business rules to the generated data models; if the models accurately reflect the business rules they are considered precise.

Maintainability/flexibility

Systems with high levels of information hiding/data abstraction are more flexible and easier to maintain. SE methodologies stress the design of functional modules with information hiding/data abstraction. Since academics teach the theory of information hiding/data abstraction, students should take these principles into account when designing a system. The IE methodology logically separates data definitions from program logic [18] and forces the separation of data and process. Within an academic setting, system maintenance is mentioned, but often neglected.

Reusability

Within two to three years, firms using IE are able to design 80% of a firm's data. Through the reuse of previously defined data and process models, applications can be developed more easily and quickly [18]. Developers using SE advocate the use of software libraries containing written and tested software modules. Either previously defined data or software libraries facilitate student software development.

Concurrent processing

Concurrency in software systems is not addressed by IE; it is supported by SE. Although most databases on microcom-

Table 2. MAJOR WORK PRODUCTS AND NOTATIONS

WORK PRODUCT	NOTATION
SE Data Flow Analysis	Data Flow Diagrams
Process Specification	Program Design Language
Data Model	E-R Diagrams
Module Definition	Structure Charts
Data Definitions	Data Dictionary
Information Hierarchy	Warnier-Orr Diagrams
Information Domain Analysis	Jackson Diagrams
IE Data Model1	E-R Diagrams
Process Specification	Process Models
Data Definitions	Data Dictionary
Process-Entity Relationships	Process Entity Matrix

[1]Once the data model is complete, the logic is used to generate procedures that process the data [31]. Automation is assumed.

puters are single-user databases, there are concurrency issues for multi-user databases. This issue is thoroughly discussed in the IS curriculum, but rarely actually encountered in student projects; students are usually working on microcomputers and tend to ignore this problem.

Performance engineering

In general, IS developers are not concerned with performance engineering issues (buffers, resource allocation); these issues are for system software developers and hardware designers.

Traceability

This property is concerned with reconciling system requirements with work products. Both SDMs have rules for traceability. In the academic environment, due to the limited scope of the project and time constraints, this is rarely an issue. However, it is fairly obvious when the student developed system does not meet the initial requirements.

Method specialization

Can the SDM be tailored to meet a system's specialized functionality? IE is based on the organization's business rules; there is built-in tailoring. SE methods can be modified; often an organization has their own system development life cycle which is a modification of the traditional SDLC. Again, in the academic environment, due to the project scope and time constraints there are few modifications to the selected SDM.

USAGE PROPERTIES

There are 5 usage properties dealing with practical issues of adopting a specific software development methodology.

System size

Both SDMs are suitable for all size applications. Due to the time constraints of the academic quarter (10 weeks) or semester (15 weeks), projects are usually small (30,000 lines of code or less). Subsets of either SDM are appropriate for an academic setting.

Applications areas

Both IE and SE are used to develop transaction processing systems where data is updated on-line. Since SE places more emphasis on code optimization, it is better suited for development of real-time systems. Using IE, code can be generated immediately after procedure design, creating prototypes and user interfaces during the construction phase. Although IE allows the user to proceed more quickly to a working system, SE also supports prototyping. Prototyping provides limited development of the system; the students can quickly achieve a working system suited to the limited time frame. Table 4 lists major application areas and the appropriate SDMs.

Automated support

Once a software development methodology is accepted, then a search for an automated tool begins. The current emphasis is on fully integrated Computer-Aided Software Engineering tools (I-CASE), which support all phases of a software development life cycle [29]. Both upper (analysis and design) and lower (coding and testing) CASE tools support SE. Developers of upper CASE tools have joint ventures with lower CASE tools developers to create the interfaces required for integration. Some developers create their own integration [30]. CADRE and Index Technologies are developers of CASE tools supporting SE methodologies. The two most prominent fully developed I-CASE tools supporting IE are developed by Texas Instruments and KnowledgeWare.

Table 3. SUMMARY OF TECHNICAL PROPERTIES

Technical Property	Information Engineering	Software Engineering
System Development Life Cycle	see Table 1	see Table 1
Major Work Products and Notations	see Table 2	see Table 2
Problem Domain Analysis	initially requires global definitions of the organization's data requirements; a super-structure for detailed work	analyzes specific application areas of the organization
Philosophy	strategic, long-term planning, behavioral	short-term immediate projects, functional
Procedure/Guidlines, Criteria. Measures	definite set of recommended per-phase procedures, lack of standards and metrics	definite set of recommended per-phase procedures, metrics and standards
Verifications	verification of SDLC stages, rarely applied in an academic settin	verification of SDLC stages, rarely applied in an academic setting
Formality	rules of the business being modeled are the rules for the data model; if the data model follows these rules, the model is considered correct	precise mathematical definitions are applied to the structured methodology designs
Maintainability/ Flexibility	flexibility - logically separates data definitions from program logic - within an academic setting maintenance is neglected	flexibility - functional modules with information hiding/data abstraction - within an academic setting maintenance is neglected
Reusability	reuse of previously defined data facilitate student software development across semesters	software libraries containing written and tested software modules facilitate student software development across semesters
Concurrent Processing	is not addressed	is supported
Performance Engineering	is not addressed	is supported and used by system software developers and hardware designers
Tracebility	rules for traceability, not often implemented within an academic setting	rules for traceability, not often implemented within an academic setting
Method Specification	based on organization's business rules; built-in tailoring, but few modifications within an academic setting	organizations modify the SDLC; few modifications within an academic setting

Table 4. APPLICATION AREAS

APPLICATION AREA	IE	SE
Transaction Processing	X	X
Real Time Systems		X
User Interface	X	
Prototyping	X	X

Table 5. SUMMARY OF USAGE PROPERTIES

Usage Properties	Information Engineering	Software Engineering
Application Areas	see Table 4	see Table 4
System Size	suitable for all size systems, in an academic setting usually small - (30,000) lines of code	suitable for all size systems, in an academic setting usually small - (30,000) lines of code
Automated Support	Integrated Computer-Aided Software Engineering tools are available	Upper and lower Computer-Aided Software Engineering tools are available and are often combined to form an integrated set of tools
Ease of Instruction	few textbooks available, often biased toward available tool, lack of case studies chronicling software development using information engineering, steep learning curve	numerous textbooks available, case studies are available chronicling software development using software engineering methodologies, steep learning curve
Maturity/Project History	mature SDM, no studies evaluating the use of IE in an academic setting	mature SDM, numerous studies involving the use of SE in an academic setting

Table 6. SUMMARY OF MANAGERIAL PROPERTIES

Managerial Properties	Information Engineering	Software Engineering
Software Development	well defined phases, milestones and deliverables - very appropriate for an academic setting	well defined phases, milestones and deliverables - very appropriate for an academic setting
Ease of Integration	evaluation of faculty talents and interests, reward system	evaluation of faculty talents and interests, reward system

Ease of instruction

There are numerous, textbooks on Software Engineering [31][19][32][12] and texts dealing with Information Engineering [8][33][15][20] are beginning to be published. Some developers of CASE products supporting IE provide, for a fee, student guidelines and materials. However, these materials are often biased, presenting IE from the tool perspective. Almost every university with either a computer science or information systems concentration has a required Software Engineering or System Analysis and Design course in the curriculum. Information Engineering courses are beginning to appear at some universities; developers of CASE products supporting the IE methodology offer courses to educators adopting their products. Since SE has been the most popular SDM, there are numerous case studies that chronicle software development using SE in both organizational and academic settings. These types of case studies are beginning to appear for IE development within an organization. Both SDMs are moderately complex with regard to notational structures and design rules. Although there are steep learning curves associated with both SDMs, there is some evidence that this does not deter students [34].

Maturity/project history

Both methodologies are mature SDMs, at least five years old, with 20 or more experience reports, 20 of which are multiple use. While there are studies [35][36][37][38] involving the use of SE methodologies in academic settings, there are none evaluating the use of IE.

MANAGERIAL PROPERTIES

There are 2 managerial properties dealing with the organizational issues of the software development method. While both of these issues are considered important, cost estimation and project staffing are not traditionally implemented in an academic setting. Project staffing is usually determined by the number of students enrolled in the course; if more man hours are needed for a project, the scope of the project probably is downsized. Cost estimation may well be the amount of time required to achieve the desired grade.

Software development organization

Both SDMs have well defined phases, milestones and deliverables that are very appropriate within the academic setting. Very specific guidelines and due dates are established; real software development structure is simulated.

Ease of integration

How easily can the methodology be integrated into the existing IS curriculum? Adopting a different methodology has ramifications throughout the entire curriculum and impacts faculty. Courses are added and/or subject matter changes; new syllabi are written. As in an organizational setting, for success everyone needs to support the same methodology. Without a faculty consensus, there is no unity within the curriculum. Evaluation of faculty talents and interests is important. Similar to IS practitioners, academics are often reluctant to alter the techniques they use. In the academic setting, adoption of different methodologies requires re-education of the participants and course revisions; many promotion and tenure models do not reward curriculum formation and faculty development.

SUMMARY

Many business schools and IS faculty are undertaking curriculum revisions and IS faculty are facing decisions about curriculum content. In order to resolve some of these dilemmas, this paper applied a framework, originally created for organizational decisions concerning SDMs, to an academic setting. This paper uses a cataloging framework [1] to compare two SDMs, Software Engineering and Information Engineering within an IS curriculum. The application of this framework can be expanded to include additional SDMs; it need not be limited to only two SDMs. Since every academic environment is unique, no decision as to the best SDM for an academic setting is made. Rather, the differences and similarities of the two SDMs are organized and presented within the framework to assist educators making these decisions. Therefore, this approach provides some guidelines for SDM selection not a recommendation for a SDM for an IS curriculum. It is clear that the framework is viable for cataloging SDMs in an IS curriculum. Using this cataloging framework, additional research is currently being conducted to determine SDM preferences among IS faculty. A future paper will report the results of the survey developed using this framework.

Some questions remain unanswered and are areas for future research: Is this the best cataloging framework for SDMs within an academic setting? Should other frameworks be used/created for use in the business IS curriculum? Is there one methodology that is better suited for business IS development? Which methodology will enable IS students to solve business problems and understand the business itself? Should/can multiple SDMs be taught, thereby exposing students to different SDMs?

Although there are numerous studies involving the use of Software Engineering in academic settings, there are no studies evaluating the use of Information Engineering in a similar setting. Additional research in this area will help answer some of the questions raised, and provide additional information to be used in conjunction with Karam and Casselman's cataloging framework.

REFERENCES

1. Karam, Gerald M. and Casselman, Ronald S. (1993) "A Cataloging Framework for Software Development Methods." *IEEE Computer*, Vol. 26, No. 2, February 1993, 34-45.

2. Keyes, Jessica (1992) "How Software is Developed Undergoing Basic Changes." *Software Magazine*, Vol. 12, No. 1, January 1992, 38-56.

3. Davis, Alan M., Bersoff, Edward H. and Comer, Edward R. (1988) "A Strategy for Comparing Alternative Software Development Life Cycle Models." *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, October 1988, 1453-1461.

4. Fichman, Robert G. and Kemerer, Chris F. (1993) "Adoption of Software Engineering Process Innovations: The Case of Object Orientation." *Sloan Management Review*, Winter 1993, 7-22.

5. Motschnig-Pitrik, Renate (1990) "A Framework for the Support of a Common Structural Level of Software-, Data Base-, and Knowledge-Based Systems." *Journal of Systems Software*, Vol. 12, No. 2, May 1990, 125-137.

6. Messenheimer, Susan and Weiszmann, Carol (1988) "Quality Software Quest." *Software Magazine*, Vol. 8, No. 2, February 1988, 29-36.

7. Norman, Ronald J. and Nunamaker, Jay F. (1989) "CASE Productivity Perceptions of Software Engineering Professionals." *Communications of the ACM*, Vol. 23, No. 9, September 1989, 1102-1108.

8. Brathwaite, Kenmore S. (1992) *Information Engineering*, Volume I, Concepts, CRC Press, Boca Raton, Florida.

9. Linger, Richard C., Mills, Harlan D. and Witt, Bernard I. (1979) *Structured Programming: Theory and Practice.* Addison-Wesley Publishing Company, Reading, Massachusetts.

10. Yourdon, Edward N. and Constantine, Larry L. (1979) *Structured Design.* Prentice-Hall, Englewood Cliffs, New Jersey.

11. DeMarco, Thomas (1979) *Structured Analysis and System Specifications.* Prentice-Hall, Englewood Cliffs, New Jersey.

12. Yourdon, Edward N. (1989) *Modern Structured Analysis*, Yourdon Press, Englewood Cliffs, New Jersey.

13. Chen, Peter P. (1976) "Entity-Relationship Model: Toward a Unified View of Data." *ACM Transactions on Database Systems*, Vol. 1, No. 1, March 1976, 9-36.

14. Codd, E. F. (1970) "A Relational Model of Data for Large Shared Data Banks." *Communications of ACM*, Vol. 13, No. 6, June 1970, 377-387.

15. Finkelstein, Clive (1989) *An Introduction to Information Engineering.* Addison-Wesley Publishing Company, Reading, Massachusetts.

16. Kerr, James M. (1991) The *IRM Imperative*, John Wiley & Sons, New York, New York.

17. Martin, Merle P. (1991) *Analysis and Design of Business Information Systems.* MacMillan Publishing Company, New York, New York.

18. Kerr, James M. (1991) "The Information Engineering Paradigm." *Journal of Systems Management*, Vol. 42, No. 4, April 1991, 28-32.

19. Pressman, Roger S. (1992) *Software Engineering.* McGraw-Hill, Inc. New York, New York.

20. Martin, James (1989) *Information Engineering.* Prentice Hall, Englewood, Cliffs, New Jersey.

21. Song, Xiping and Osterweil, Leon J. (1992) "Toward Objective, Systematic Design-Method Comparisons." *IEEE Software*, Vol. 9, No. 3, May 1992, 43-53.

22. Gilb, Thomas (1977) *Software Metrics.* Winthrop Publishers, Inc., Cambridge, Massachusetts.

23. Halstead, Maurice H. (1977) *Elements of Software Science.* Elsevier, New York.

24. Elshoff, James L. (1984) "Characteristic Program Complexity Measures." Proceedings of the 7th *International Conference on Software Engineering*, March 1984, Orlando, Florida, 288-293 IEEE Computer Society Press, Los Angeles, California.

25. Weyuker, Elaine J. (1988) "Evaluating Software Complexity Measures." *IEEE Transactions on Software Engineering*, Vol. SE-14, No. 9, September 1988, 1357-1365.

26. *IEEE Transactions on Software Engineering*, Vol 18, No. 11, November 1992, Special Issue on Software Measurement Principles, Techniques and Environments, eds. Selby, R. W. and Torii, K.

27. Boehm, Barry. W. (1984) "Verifying and Validating Software Requirements and Design Specifications." *IEEE Software*, January 1984, 75-88.

28. Karimi, Jahangir, and Konsynski, Benn R. (1988) "An Automated Software Design Assistant." *IEEE Transactions on Software Engineering*, Vol. 14, No. 2, February 1988, 194-210.

29. Chen, Minder and Norman, Ronald J. (1992) "A Framework for Integrated CASE." *IEEE Software*, Vol. 9, No. 2, March 1992, 18-22.

30. Steinberg, Geoffrey and Baram, Giora (1992) "An Investigation of CASE Software Integration." *Journal of Systems Management*, Vol. 43, No. 9, September 1992, 20-22.

31. Fairley, Richard (1985) *Software Engineering Concepts*, McGraw-Hill Co., New York, New York.

32. Schach, Stephen R. (1990) *Software Engineering*. Irwin, Homewood, Illinois.

33. Brathwaite, Kenmore S. (1992) *Information Engineering*, Volume II, Analysis and Administration, CRC Press, Boca Raton, Florida.

34. Granger, Mary J. and Pick, Roger Alan (1991) "The Impact of Computer-Aided Software Engineering on Student Performance" *Proceedings of the Twenty-Second Special Interest Group on Computer Science Education (SIGCSE)*, March 1991, San Antonio, Texas, 62-72.

35. Aukerman, Richard; Schooley, Robert; Nord, Daryl and Nord, Jeretta. (1989) "The Importance of Selected Systems Analysis and Design Tools and Techniques as Determined by Industry Systems Analysts and University Educators." *SIGCSE Bulletin*, Vol. 21, No. 3, September 1989, 30-34.

36. Joyce, Daniel (1987) "An Identification and Investigation of Software Design Guidelines for Encapsulation Units." Doctoral Dissertation, Temple University 1987.

37. McKeeman, William M. (1987) "Experience with a Software Engineering Project Course." *IEEE Transactions on Software Engineering*, Vol. SE-13, No. 11, November 1987, 1182-92.

38. Selby, Richard W., Basili, Victor R., & Baker, F. Terry (1987) "Cleanroom Software Development: An Empirical Evaluation." *IEEE Transactions on Software Engineering*, SE-13, No. 9, September 1987, 1027-37.

Author's Biography

Mary J. Granger is Assistant Professor of Management Science in the School of Business and Public Management at The George Washington University. She received her Ph.D degree in Information Systems from the University of Cincinnati. Her current research interests focus on Information Systems curriculum development, CASE and international Information Systems issues.

Author's Biography

Ross A. Malaga received his Masters of Science in Information Systems from The George Washington University and is currently pursuing his PhD in Information Systems at George Mason University. Mr. Malaga is currently employed by Ogden Government Services as an Information Engineer.

Subscribe!

Subscribe to the Journal of Information Systems Education and you'll get far more than this valuable journal! Because the Journal of Information Systems Education is published by EDSIG, the DPMA's special interest group for education. And as a subscriber, you'll get:

- Four quarterly issues
- Special discounts on EDSIG sponsored activities
- Networking opportunities with others in your profession
- Access to members via Bitnet/Internet
- A forum for discussion of IS teaching and training topics
- The ability to share information on student chapters

Yes! Start my subscription today. I've enclosed a check for $35, payable to the Journal of Information Systems Education, with this completed coupon. (Mail to: John Corrigan, COMPUTERWORLD, 375 Cochituate Road, Framingham, MA 01701.)

Name: _____

Company or Institution: _____

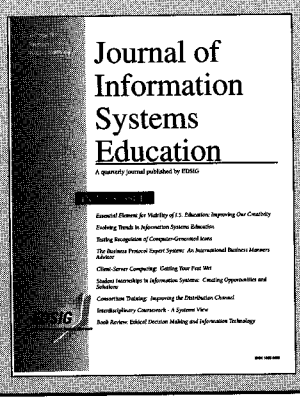
Address: _____

City/State/Zip: _____

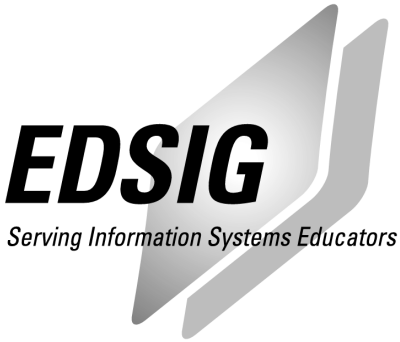
Phone: _____

Fax: _____

E-mail: _____



Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

Copyright ©1994 by the Information Systems & Computing Academic Professionals, Inc. (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to the Editor-in-Chief, Journal of Information Systems Education, editor@jise.org.

ISSN 1055-3096