Journal of International Information Management

Volume 6 | Issue 1 Article 6

1997

Maturity Model Linkages Between Software Development Teams, Users and Quality

Eugene G. McGuire
The American University

Maturity Model Linkages Between Software Development Teams, Users and Quality

Eugene G. McGuire
The American University

Abstract

Information Systems literature has shown clear linkages between the selection of software development methodologies, the actions of software development teams, overall product quality, and user satisfaction. Rarely are these linkages emphasized, however, in current research on process improvement. This paper examines the convergence of these areas and discusses current research designed to further explore these linkages within the Capability Maturity Model (CMM) framework.

Introduction

Many current studies which focus on process improvement issues overlook or minimize the importance of fundamental relationships between features of different software development methodologies, task and relationship behaviors of software development teams, overall, measurable product or system quality, and finally, user satisfaction. These linkages, however, can provide key insights into fundamental organizational issues affecting this process as well as into underlying behavioral issues related to the people involved on the development process: software development teams and end users. This paper illustrates several of these linkages and discusses ongoing research designed to track changes in behavioral and work attitudes among both technical personnel (software development teams) and end users as organizations undertake software and systems process improvement through structured approaches such as the Capability Maturity Model and related assessment models. Of particular interest are the prototype systems engineering and people management capability maturity models being developed by the Software Engineering Institute (SEI).

The vast body of Total Quality management (TQM) literature repeatedly emphasizes key concepts such as a concentrated focus on customer involvement and satisfaction, statistical quality control, and improvement of process until zero-defect goals are reached. Implementation of TQM concepts in Information Systems (IS) environments, however, can often be difficult. This difficulty can be attributed to the inherent differences between the intellectually intensive process of developing systems as opposed to the mechanically intensive process of manufacturing products (where TQM often excels). Other, perhaps more realistic, viewpoints based on systems development activities address the logic and flow of the traditional systems development process and find flaws that can be readily addressed by quality concepts.

Software Development Methodologies

There are many different life-cycle models in use today, including the traditional waterfall based model of Royce (1970), the Spiral Model (Boehm, 1988), the use of CASE environments, prototyping, and formal methods having mathematical specification techniques. These models are used in conjunction with the procurers' requirements and standards such as those proposed by ANSI, IEEE, DoD, and American Institute of Aeronautics and Astronautics (AIAA).

Royce (1970) proposed a stage-based model, the waterfall model, to facilitate the development of large-scale software systems. This approach was an attempt to characterize the various phases of a software development project and to assist in the management of such projects. This model assumed that the various stages were discrete and that each stage's output represents an input for the next one. However, as development can now be seen as a continuous function, new models of development have been suggested.

Modern complex systems require a more effective development process than the traditional waterfall systems development life cycle. The waterfall approach emphasizes the completion of one life cycle stage before proceeding onto the next stage. Unfortunately, the side effects of this inflexible development process include freezing customer input early in the development process, limiting continued customer involvement, and testing modules too late in the cycle. As a result, maintenance costs become uncontrollable because systems are developed that do not satisfy customer requirements and continue to be error-ridden.

In the past, the popular systems development methods such as Structured Systems Analysis and Design (SSAD) (Yourdon, 1989) and Jackson System Development (JSD) (Cameron 1989) have applied the traditional life cycle approach (or similar variations) to developing systems. The major activities in the life cycle method most often include the specification, analysis, design, implementation, installation, and system maintenance (Yourdon & Constantine, 1979). The life cycle method is relatively inflexible because each activity is significantly completed before another activity is begun. This approach is acceptable in these methodologies because they do not include a formal verification process which often requires an iteration of previous activities.

There are other types of system development methods such as prototyping that offer a dynamic approach to the systems development process (Naumann & Jenkins, 1982). Prototyping has become a popular alternative to the static life cycle method of systems development. Even though such methods allow for user feedback during the design process, they do not provide the means for formal verification of the design.
It has been shown that these methods are quite easy to apply during the systems development process. Unfortunately, there is no guarantee that the functionality of the implemented system is consistent with its specification. As a result, unplanned maintenance expenses are incurred to fix errors in the system design.

The Spiral Model (Boehm, 1988) and prototyping are two development models which utilize the iterative approach of developing systems incrementally so they will be acceptable to the user. In the Spiral

Model, the process ends with the development of a system which adequately addresses the needs of the users. The prototyping approach utilizes the prototype as a definition of the user requirements. A system design can then be based on this prototype. The formal transformation model utilizes formal, mathematical-based methods for requirements specifications and then transforms these specifications to programs.

The spiral development process reflects the actual iterative progress of the development of a system (Mills, 1986). Instead of performing a static set of activities in a predefined order, the spiral development process allows any one of three activities (Mills, 1987): investigation, specification, or implementation to be performed in any time sequence.

- The investigation activity is a discovery process that includes assessing the system's feasibility, and gathering facts about its behavior.
- The specification activity includes analyzing and designing system components. During the specification activity, each system design and design modifications are formally verified for correctness.
- The implementation activity transforms the system design into an operational system. Statistical usage testing is performed to ensure that any errors that were not discovered during design are uncovered and can be corrected.

Any of these activities can be performed iteratively and in any time sequence. For example, during a specification activity, it may be discovered that more information about the system is needed to complete a design. An investigation activity is then performed to gather additional facts. During the implementation activity, if an error is discovered, the system design is modified and verified for correctness before the usage testing is continued.

Shortcomings of Methodologies

All of these methods of software development, to varying degrees, have a common shortfall -- the absence of process control. These models clearly defined what should be done in every phase and what the resulting product of that phase should be; however, they did not mention how each phase's performance should be controlled in order to increase quality through measurement and management.

A software process movement emerged in the mid-1980s when shortcomings in managing software development processes were recognized as impediments to improving software reliability and quality. One model of software process maturity that has received considerable attention is the Capability Maturity Model (CMM) developed by the Software Engineering Institute (SEI) at Carnegie Mellon University in Pittsburgh.

The original SEI software process maturity model was developed to assess large software houses developing technical systems for the U.S. Air Force and other defense agencies. Software process

improvement provides guidelines for an organization to employ a structured approach to strategically improve all aspects of a software development operation. Organizations previously would often attempt to improve performance in an ad-hoc manner by using technology driven approaches instead of addressing root problems. For example, organizations would often assign more staff or more tools to an already late and over-budget development project rather than addressing the reasons why the project had problems.

In contrast, in a mature organization (as measured under the CMM framework), managers use metrics to quantitatively measure the quality of a software product. Schedules and budgets are based on historical performance and are realistic; the predicted results for product cost, schedule, functionality, and quality are routinely achieved. Based on the predictability of the project results, reliable management decisions can be made about tradeoffs among these outcomes. Management can also more clearly define relationships concerning roles and responsibilities between software developers and users.

Some important goals of a more clearly defined link between software developers, users, and quality are:

- clearer communication between customer and supplier (in all phases),
- measurement of existing process' attributes and establishment of standards,
- continuous improvement effort,
- shift of focus from testing to requirements specification and design,
- shift from an artistic belief about software development to an engineering one.

These goals recognize the importance of communication between customer and software developer as well as between professionals with different skills. It is expected that by establishing formal procedures for communication, performance will be enhanced and problems diminished. Furthermore, the establishment of informal meetings, under the umbrella of quality and process control, will enhance the understanding between professionals with different educational backgrounds, as well as help in the dissemination of information which would not happen otherwise. In addition, such informal meetings will foster the team spirit between members of an organization.

The establishment of a software metrics program will provide the necessary insight in the process and product attributes. It will also become the foundation of an improvement program. These goals propose a formal mechanism for continuous improvement. The advantage of such a mechanism is that the need for the process to be always considered as under evaluation is recognized and thus the performance is new allowed to deteriorate.

Another aim of this approach is to move the emphasis of the development from being an artistic expression to one of engineering. Some research (Cougar et al, 1978; 1980; 1990), however, shows that the single most important motivator for data processing professionals is the job itself. Thus a shift

or perceived removal of the creative aspect from the software developer's work function could lead to a decrease in performance. One solution to this possibility is to shift the creativity of software developers from the software product to the software process thereby focusing on ideas for improving the process and the resulting final product as well.

In addition, self-managed teams appear to be evolving at a rapid pace as a result of TQM and reengineering initiatives in many organizations. The literature shows many studies which demonstrate that extraordinary productivity results when people are placed in self-managed teams in highly constrained environments. There is a substantial literature relating tales of "heroic" operations working against impossible deadlines with minimal resources, thereby becoming extraordinarily motivated and sometimes flouting the expectations of the mature parent organization. Some of these projects were poorly structured but, nevertheless, succeeded as a result of the personalities of the leaders (Gardiner, 1990; Guterl, 1984; Kidder, 1981). It appears clear that software developers can exercise considerable creativity and autonomy within self-managed teams that are operating under the CMM framework if they accept the possibility and potential of creativity within structured boundaries.

Information Technology Total Quality Barriers

Although Total Quality Management (TQM) has had a considerable impact on management practices and the work of employees during the last decade, signs have emerged indicating that TQM's presence and impact may be on the wane. A growing number of companies are stepping away from their already-established TQM programs (Fisher, 1994; Mathews, 1993) while other organizations are opting not to take on the daunting challenge of implementing TQM (Bleakley, 1993). In the wake of a number of studies reporting implementation problems and disappointing results (Fuchsberg, 1992), there are reports of management disillusionment with what one report terms "totalled quality management" (Mathews, 1993).

Barriers to quality improvement include ten barriers identified in the literature (Scandura & Stewart, 1993). These ten barriers are specifically the absence of: 1) employee empowerment; 2) management's listening to employees; 3) employee training; 4) mutual trust between manager and employee; 5) team building; 6) quality-related goals for employees; 7) a performance appraisal system that measures and rewards quality improvement; 8) reasonable workload expectations; 9) a stable and relatively certain work environment; and 10) employee acceptance (nonresistance to the TQM initiative.

It is certainly possible that these barriers have become more prominent as TQM initiatives have expanded beyond the original manufacturing environments into the intellectual environments of information technology and software development. The role of information technology and associated organizational attitudes towards it have dramatically changed in a relatively short period of time. Until the late 1970s, most social science activities related to technology and innovation were limited to the studies of how technological innovations diffuse and are adopted (Bower, 1937; Ryan & Gross, 1943; Rogers, 1962; Gould, 1969).In the late 1970s, however, technology was increasingly seen as a corporate strategic issue (Drucker, 1973, Fahey & King, 1977; Fusfeld, 1978; Ansoff, 1979) practically exploding in both number and in scope during the 1980s (Ford & Ryan, 1981; Galbraith, 1984; Abetti, 1985;

Drucker, 1985; Mitchell, 1985; Quinn, 1985, 1986; Foster, 1986; Friar & Horwitch, 1986, Hamilton, 1986; Ansoff, 1987; Betz, 1987; Ford, 1988). From then on, change was related to technology in the strategic literature almost as a matter of course, and thus became a strategic issue which appeared at the top of the strategic agenda (Morton, 1991; Stokes, 1991; Hammer & Champy, 1993; Allen & Morton, 1994).

In addition, empirical work in recent years has paid much attention to the political/organizational aspects of information systems development (Keen, 1981; Markus, 1983; Newman & Rosenberg, 1985; Hirschheim, Kline & Newman, 1987; Kling & Scacchi, 1982; Kling 1987). The organizations into which information systems are introduced are seen as political orders, with established distributions of power which may be disrupted or confirmed by the design of the technology (Pettigrew, 1973; Markus, 1981; Markus & Pfeffer, 1983). Moreover, writers such as Keen and Gerson (1977), Franz and Robey (1984), Kling (1987), and Markus and Bjorn-Andersen (1987) have pointed out the political nature of the development process itself. Kling (1987, p. 312) explicitly includes a political model as part of the assumptions for his "web" model of computing in organizations.

Political models take account of conflict but are concerned with the way in which conflict is structured by the existing organization, and the role of power in conflict resolution. Users and systems specialists have interests which they pursue through the development process. Political tactics such as bargaining and negotiations are relevant to conflict resolution. If these are not successful, information systems development can become a power struggle with a win-lose outcome determined by which side has the most resources. Software developers and users, because of organizational culture and conflicting pressures, may not be easily linked together in common quality-oriented goals.

End Users and Usability

End-users of computer applications have discovered one benefit of usability: increased productivity. Increased productivity results from savings in time to train, learn, use, and migrate between products -- with decreased error rates. Also, product usability increasingly appears in the end-users' definition of quality. For example, a recent survey of end users of computer systems ranked ease-of-learning and use as the number one differentiator of product quality (Snyder, 1991). It could be said that end-users have discovered that usability is a link among productivity, quality, and technology.

User-centered development is not new (for example, Gould & Lewis, 1983; Norman, 1988; Norman & Draper, 1986; Wixon & Whiteside, 1985), but the application of the principles is now more critical than ever before. These sources demonstrate the need for and the utility of systems designed with usability as a key component.

Gould and Lewis (1983) proposed four principles for the practice of usability engineering: early focus on users, integrated design of the interface, early and continual testing, and iterative design. Gould, Boise and Lewis (1993) summarized the effect of the approach and suggested future directions. As Gould, et al (1991) indicated, a customer-centered usability approach works, but there are some problems. The primary problem is that usability does not have priority or parity with cost and schedule

in most organizations. Again, it appears that a structured process improvement approach such as the CMM offers not only guidelines for quantitatively evaluating various processes but also for clarifying issues of quality related to usability by emphasizing better working relationships between software developers and users.

Current Research

Most organizations have goals to reduce costs and intervals. It should be noted that the SEI CMM does not directly focus on reducing costs or intervals. This model instead focuses on the best practices of projects that develop high quality software. Models such as the CMM should contribute toward lower costs and reduced intervals because they result in high quality processes. These processes are capable of predictable results (producing high-quality software products) which in turn help with the planning and management of projects.

Two variations on the CMM are the Systems Engineering Capability Maturity Model (SE-CMM), Version 1.0 (December 1994) and the People Capability Maturity Model (P-CMM), Version 0.3 (April 1995). These adaptations of the original CMM seek to focus the organization's resources on the whole systems engineering process and the human resources aspects of the processes respectively. By extending the initial CMM framework to cover systems engineering and human resource factors the SEI is attempting to address problems that fall outside the software domain. Both of these new models, particularly the P-CMM include substantive discussion of how people, technical and end-user, are instrumental in the production of quality software and systems.

Most improvement programs to date have emphasized process or technology and not people. People management practices in many organizations, despite a significant amount of literature addressing people-related issues, do not address people issues in a systematic and structured manner. In addition, most managers are untrained or inadequately trained in implementing corrective solutions once people problems are identified.

Current research involves preparing a gap analysis of current and desired states of different human resource issues. The term "gap" is defined as the difference between the actual state of a process or practice within a project and the desired state or practice as derived from the SEI CMM profile. This research is being conducted at a large multi-national organization with 20 software development teams working on different aspects of a large multi-phase, multi-year project. These 20 different teams operate within the same overall organizational structure but are at different levels of process maturity because of the nature of their particular projects.

The 20 teams are being surveyed over a period of time on the key process areas identified in the CMM and the P-CMM for each maturity level.The key process areas in the CMM are:

Level 2 -- requirements management; software project planning; software project tracking and oversight; software subcontract management; software quality assurance; software configuration management.

Level 3 -- organization process focus; organization process definition; training program; integrated software management; software product engineering; intergroup coordination; peer reviews.

Level 4 -- quantitative process management; software quality management

Level 5 -- defect prevention; technology change management; process change management

The key process areas in the P-CMM are:

Level 2 -- people management values development; staffing; performance management; training and career development; compensation and reward; participatory culture; work environment.

Level 3 -- people management planning; knowledge and skills analysis; competency development; competency-based practices; team building.

Level 4 -- quantitative people management; organizational competency management; organizational performance alignment.

Level 5 -- continuous people management improvement; people management innovation.

The goal of this research is to develop a matrix of team characteristics and a prioritized list of "gaps" for each team between the current and desired states of the above factors. Identification of these gaps will assist the organization in prioritizing areas needing improvement for each team and enable them to compare attitudes of the team members with the appropriate key process area in the appropriate maturity level.

References

Abetti, P.A. (1985). Milestones for managing technological innovation, ***Planning Review,*** 13 *(2)*, pp. 18-23.

Allen, T.J. & Morton, M.S. (1994). ***Information technology and the corporation of the 1990s: Research studies.*** Oxford University Press: NY.

Ansoff, H.I. (1979). ***Strategic management***, Macmillan: London.

Ansoff, H.I. (1987). Strategic management of technology, ***Journal of Business Strategy***, 7, pp. 28-39.

Betz, F. (1987). ***Managing technology: Competing through new ventures, innovation, and corporate research***, Prentice Hall: Englewood Cliffs, NJ.

Bleakley, F.R. (1993, July 6). Many companies try management fads, only to see them flop, ***The Wall Street Journal***, pp. A1, 6.

Boehm, B.W. (1988, May). A spiral model of software development and enhancement, ***IEEE Computer***, pp. 61-72.

Bower, R.V. (1937). The direction of intra-societal diffusion, ***American Sociological Review***, *2*, pp. 826-836.

Cameron, J. (1989). ***JSP and JSD: The Jackson approach to software development***, IEEE Computer Society Press.

Cougar, J.D. & Zawicki, R.W. (1978, Sept.). What motivates DP professionals," ***Datamation***, , pp. 116-123.

Cougar, J.D. & Zawicki, R.W. (1980). ***Motivation and management of computer personnel***, NY: Wiley.

Cougar, J.D. (1990, Jan. 15). Motivating analysts and programmers, ***Computerworld***, pp. 73-76.

Drucker, P.F. (1973). ***Management: tasks, responsibilities, practices***. Harper & Row: NY

Drucker, P.F. (1985). ***Innovation and entrepreneurship: Practice and principles***, Harper & Row: NY.

Fahey, L. & King, W.R. (1977). Environmental scanning for corporate planning, ***Business Horizons***, 20 (4), pp. 61-71.

Fisher, L. (1994). Total quality: Hit or myth? *Accountancy*, vol. 113, pp. 50-51.

Ford, D. (1988). Develop your technology strategy, ***Long Range Planning***, 21 *(5)*, pp. 85-95.

Ford, D. & Ryan, C. (1981). Taking technology to the market, in R.A. Burgelman & M.A. Maidique, Eds., ***Strategic management of technology and innovation***, Irwin: Homewood, IL.

Foster, R. (1986). ***The attacker's advantages***, Summit Books: NY.

Franz, C.R. & Robey, D. (1987). Strategies for research on information systems in organizations: Research purpose and time frame, in R. Boland, Jr. & R.A. Hirschheim, ***Critical issues in information systems research***, Wiley: Chichester, pp. 205-225.

Friar, J. & Horwitch, M. (1986). The emergence of technology strategy. A new dimension of strategic management," in M. Horwitch, (Ed.), ***Technology in the modern vorporation: A strategic perspective***, Pergamon Press: NY.

Fuchsberg, G. (1992, May 14). Quality programs show shoddy results, ***The Wall Street Journal***, pp. B1, 9.

Fusfeld, A.R. (1979). How to put technology into corporate planning, in R.A. Burgelman & M.A. Maidique, Eds., ***Strategic management of technology and innovation***, Irwin: Homewood, IL.

Galbraith, J.R. (1984). Designing innovative organizations, in R.B. Lamb, (Ed.), ***Competitive strategic management***, Prentice Hall: Englewood Cliffs, NJ.

Gardiner, K.M. (1990). D esign: Organization and measurement," chapter in the ***Handbook of design management,*** Basil Blackwell: Oxford, pp. 155-166.

Gould, J.D., Boies, S.J., & Lewis, C.H. (1991). Making usable, useful, productivity enhancing computer applications," ***Communications of the ACM***, 34 *(1)*, pp.74-85.

Gould, J.D. & Lewis, C.H. (1983). Designing for usability -- Key principles and what designers think, ***Proceedings of the 1983 Computer-Human Interaction Conference***, New York, pp. 50-53.

Gould, P.R. (1969). ***Spatial diffusion as a spatial process***, University of Chicago Press: Chicago.

Guterl, F. (1984, Dec.). Design case history: Apple's Macintosh," ***IEEE Spectrum***, Vol. 21, No. 12.

Hamilton, W.F. (1986). Corporate strategies for managing Eeerging technologies," in M. Horwitch, (Ed.), ***Technology in the modern corporation: A strategic perspective***, Pergamon Press: NY.

Hammer, M. & Champy, J. (1993). ***Reengineering the corporation: A manifesto for business revolution***. HarperBusiness: NY.

Hirschheim, R. & Newman, M. (1988). Information systems and user resistance: Theory and practice, ***The Computer J.***, 31, pp. 398-408.

Hirschheim, R., Klein, H., & Newman, M. (1987). A social action perspective on information systems development, ***Proceedings of the Eighth ICIS Conference***, Pittsburgh, PA, pp. 45-56.

Keen, P.G.W. (1981). Information systems and organizational change, ***Communications of the ACM***, 24, pp. 24-33.

Keen, P.G.W. & Gerson, E.M. (1977). The politics of software systems design, ***Datamation***, 11, pp. 80-84.

Kidder, T. (1981). ***The soul of a new machine***, Little, Brown, and Company: Boston, MA.

Kling, R. (1987). Defining the boundaries of computing across complex organizations, in R. Boland, Jr. & R.A. Hirschheim, ***Critical issues in information systems research***, Wiley: Chichester, pp. 307-362.

Kling, R. & Scacchi, W. (1982). The web of computing: Computing technology as social action, in ***Advances in Computing***, 21, Academic Press, New York.

Markus, M.L. (1981). Implementation politics: Top management support and user involvement, ***Systems, Objectives, Solutions***, 1, pp. 203-215.

Markus, M.L. (1983). Power, politics, and MIS implementation, ***Communications of the ACM***, 26, pp. 430-444.

Markus, M.L. & Pfeffer, J. (1983). Power and the design and implementation of accounting and Control Systems, ***Accounting, Organizations and Society***, 8, pp. 205-218.

Markus, M.L. & Bjorn-Andersen, N. (1987). "Power Over Users: Its Exercise by System professionals, ***Communications of the ACM***, 30, pp. 498-504.

Mathews, J. (1993, June 6). Totalled quality management consultants flourish helping firms repair the results of a business fad, ***The Washington Post***, p. H1.

Mills, H., Linger, R., & Hevner, A. (1986). ***Principles of information systems Analysis and design***, Academic Press: NY.

Mills, H. (1987). Box structured information systems, ***IBM Systems Journal***, Vol. 26, No. 4, pp. 395-413.

Mitchell, G.R. (1985). New approaches to the strategic management of technology, ***Technology in Society***, 7, pp. 227-239.

Morton. M.S. (1991). ***The corporation of the 1990s; Information technology and organizational transformation***. Oxford University Press: NY

Naumann, J.D. & Jenkins, M. (1982). Prototyping: The new paradigm for systems development, ***MIS Quarterly***, September 1982, Vol. 6, No. 3, pp. 29-44.

Newman, M. & Rosenberg, D. (1985). System analysts and the politics of Ooganizational control, ***Omega***, 13, pp. 393-406.

Norman, D.A. (1988). ***The design of everyday things***. Doubleday: NY.

Norman, D.A. & Draper, S.W. (Eds.) (1986). ***User-centered system design: New perspectives on human-computer interaction***, Hillsdale, NJ: Erlbaum Associates.

Pettigrew, A. (1973). ***The politics of organizational decision-making***, Tavistock: London.

Quinn, J. (1985). How companies keep abreast of technological change, ***Long Range Planning,*** 18 *(2),* pp. 69-76.

Quinn, J.B. (1986). Innovation and strategy: Managed chaos, in M. Horwitch, (Ed.), ***Technology in the modern corporation: A strategic perspective***, Pergamon Press: NY.

Rogers, E.M. (1962;1983). ***Diffusion of innovations, 3rd. ed.*** (first published in 1962), The Free Press: NY.

Royce, W.W. (1970). Managing the development of large software systems: Concepts and techniques, ***Proceedings of WESTCON***, California. (Also available in ***Proceedings of ICSE 9.*** Los Alamitos, California: IEEE Computer Society.)

Ryan, B. & Gross, N. (1943). Diffusion of hybrid seed corn in two Iowa communities, ***Rural Sociology,*** *8*, pp. 15-24.

Scandura, T.A. & Stewart, K.A. (1993). An employee and managerial perspective on quality/productivity improvement programs: Benefits, costs, obstacles, and requirements for success, ***Productivity and Quality Management Frontiers***, vol. 4, pp. 1077-1086.

Snyder, K.M. (1991). ***A guide to software usability***, IBM Publication SC26-30000, New York: International Business Machines Corp.

Stokes, S.L. (1991). ***Controlling the future: Managing technology-driven change***. QED Information Services, Inc.: Wellesley, MA.

Yourdon, E. & Constantine, L.L. (1979). ***Structured design: Fundamentals of a discipline of computer program and systems design***, Prentice Hall: Englewood Cliffs, NJ.