BEHAVIORAL PERSPECTIVES ON THE IMPACTS OF TECHNOLOGY ON I/S PROTOTYPING

Jay G. Cooprider
John C. Henderson

January 1989

CISR WP No. 189
Sloan WP No. 3040-89-MS

Center for Information Systems Research
Massachusetts Institute of Technology
Sloan School of Management
77 Massachusetts Avenue
Cambridge, Massachusetts, 02139

BEHAVIORAL PERSPECTIVES ON THE IMPACTS OF TECHNOLOGY ON I/S PROTOTYPING

Jay G. Cooprider
John C. Henderson

January 1989

CISR WP No. 189
Sloan WP No. 3040-89-MS

Center for Information Systems Research
Sloan School of Management
Massachusetts Institute of Technology

Behavioral Perspectives on the Impacts of Technology on I/S Prototyping

Jay G. Cooprider
John C. Henderson

ABSTRACT

Builders of information systems face an increasingly turbulent and dynamic environment. This unstable environment is forcing I/S designers to be increasingly flexible and responsive to both a rising demand for and a variety of systems. Prototyping has received a great deal of attention as a design methodology that addresses these issues.

Most support technologies for prototyping environments focus on increasing the efficiency of an individual system builder. This paper proposes that a broader perspective is required to fully assess the potential performance impacts of support technology on the prototyping process. In building this perspective, frameworks for prototyping processes, support technology and development performance are presented. Prototyping is characterized from a behavioral perspective as a combination of cognitive, social and organizational processes -- each of which must be addressed in assessing impacts. Support technology is characterized by production, coordination and organizational dimensions, each affecting prototyping processes and performance in different ways. To assess the impacts of support technology, measures are suggested for prototyping processes and products from task, social and strategic perspectives. It is proposed that the primary determinant of the performance impacts is the fit between the prototyping processes and the support technology used. By combining a functional model of support technology with behavioral perspectives of the prototyping process, a more complete understanding of the impacts of technology on the performance of prototyping is obtained.

Behavioral Perspectives on the Impacts of Technology on I/S Prototyping

1.0 Introduction

The job of information system (I/S) builders is becoming more difficult. They must deal with more sophisticated end users [74, 72], demands for increased productivity [13, 16, 56], and proposals for a wide array of development methods, techniques and technologies [2, 36, 66]. Their environment is growing increasingly dynamic and turbulent [43] as their organizations confront more intense competitive pressures (from both domestic and foreign sources), changing regulatory controls, shrinking product life cycles, shifting management and control strategies, and growing uncertainties about their rapidly evolving technology base. These factors are making it increasingly difficult for today's I/S builder to meet the performance expectations of management and the user community.

Prototyping has received a great deal of attention in recent years as a design methodology capable of addressing many of these issues. It has been suggested that prototyping enables developers to build systems more quickly [8, 62]; increase user involvement, utility and satisfaction [20, 3, 82]; respond to changing environments and requirements [81]; decrease maintenance costs [8, 82]; lower technical risk levels [8, 24]; reduce the number of design defects [52]; and reduce the level of application uncertainty [3, 81]. Many view prototyping as the methodology that represents the future of software development [2, 8, 61].

Past researchers have proposed many factors as affecting the performance of prototyping projects. The skills of the I/S builder [65], the strength of the project leader [77], the experience of the system builder [8], the characteristics of the application [8, 65], and top management support [14] have each been proposed as determinants of prototyping success. A major thread through a great deal of the prototyping literature, however, is that the availability of appropriate tools and technologies is a major enabler of success for prototyping projects [8, 20, 26, 65, 81].

However, it is now being realized that technology per se is not the answer to successful rapid prototyping. As Lipp [58] states in his book on the prototyping approach to systems development:

> Many DP organizations have approached problem resolution from a 'tool-oriented' perspective, assuming that if enough sufficiently sophisticated tools are "thrown" at the problem it will be solved. ... [This approach is] simply providing a better weapon with which to kill ourselves [58, p. 147].

Technology *can* have a major impact on the performance of I/S builders using prototyping, but the technology must complement the processes used by builders during a project [24, 25]. There are many reasons that Lipp [57] and others [19] have not found supporting technology to have a significant effect on development performance [39]. A major reason for this lack of effect, however, is a mismatch between the capabilities of the technology used and the fundamental processes of the system builder. It is the premise of this paper that large impacts on prototyping performance are possible only when supporting technologies are properly matched with the fundamental processes of the system prototyper. This "match" between technology and process will be referred to as "fit". In doing so, fit is regarded as a moderating relationship [83] -- technology "moderates" (or enhances) the relationship between process and performance. Specifically, though the prototyping processes and support technologies both affect prototyping performance directly, the *fit* between process and technology will be the significant determinant of performance. Figure 1 illustrates this relationship.

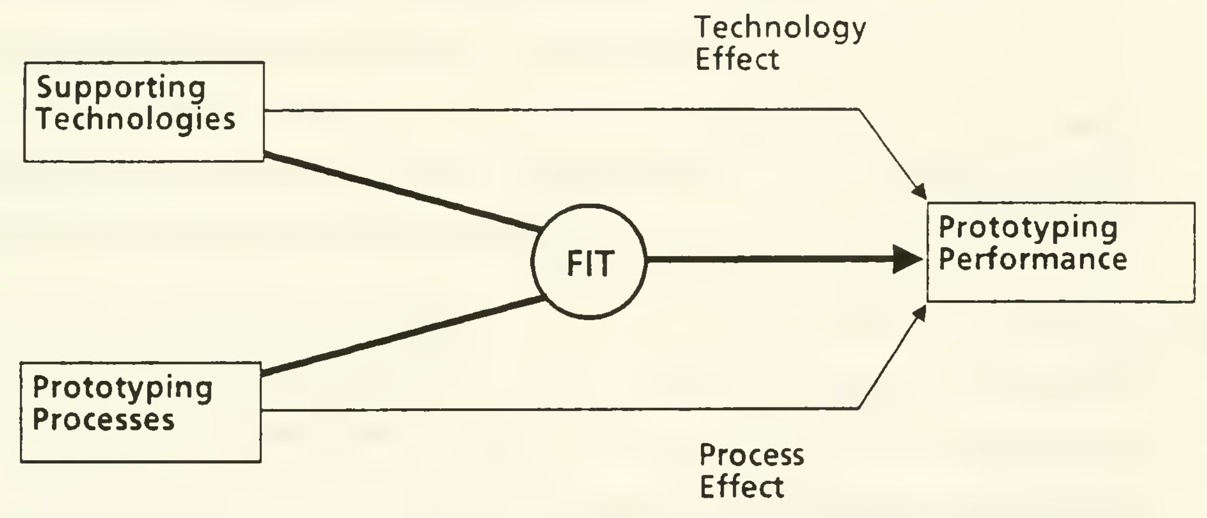

Figure 1 -- The Impact of Support Technology on Prototyping Performance

A closely related issue involves measuring the performance of prototyping projects. In order to assess the impact of support technology on the prototyping process, it is obviously important to be able to measure that impact. However, traditional measures of I/S development performance (such as number of lines of code produced by a given amount of effort) distort the overall performance picture and cause managers to overlook important productivity factors [56, 53]. To overcome this, we present a framework for measuring prototyping performance. This framework will match performance measures to the perspectives of the prototyping processes and support technologies that we will establish.

This paper, then, investigates the performance impacts of support technology on the prototyping process by: 1) describing the fit between the dimensions of technology and the important perspectives of the prototyping processes, and 2) discussing the various performance impacts that this fit implies. Section 2 examines the behaviors of I/S builders during prototyping and establishes behavioral perspectives on prototyping processes that will be used as a basis for discussing the relationships between technology and performance. Section 3 characterizes support technologies for prototyping, identifying the dimensions that benefit the system prototyper. Section 4 then builds a framework for evaluating the performance of prototyping projects. This section identifies key dimensions of I/S performance, provides examples of useful measures, and discusses potential performance impacts of various dimensions of support technology on the different perspectives of the prototyping process. Finally, Section 5 offers final conclusions and recommendations for further research.

2.0 Behavioral Perspectives on the Prototyping Process

In the context of this paper, a "process" is a systematic approach to the creation of a product or the accomplishment of a task [69]. A number of characterizations of the prototyping process have been given in the literature. Naumann and Jenkins [65] characterize prototyping as a four-step procedure:

1. Identify the User's Basic Information Requirements
2. Develop a Working Prototype
3. Implement and Use the Prototype System
4. Revise and Enhance the Prototype System

Many other authors have presented similar descriptions. For example, Scharer [77] describes the prototyping process as:

1. Preliminary Fact-finding
2. Pregeneration Design
3. Prototype Generation
4. Prototype Refinement Iterations

In general, these characterizations of the prototyping process describe prototyping as a set of phases that are terminated by the existence of an artifact -- milestones, in effect. As such, they share a fundamental problem with the traditional "waterfall" method of software development [10] that they are trying to augment [48] or replace [32, 64]. These "milestone" process descriptions do not provide insight into the actual design behaviors or the processes that actually determine the success of the prototyping effort [24]. If a design process model only depicts idealized processes that do not accurately map to actual prototyping behavior, then the supporting technologies that are applied to these idealized processes will not match the way that prototypers really work. Applying technology in this way may actually impair prototyping performance rather than enhance it.

In order to realize the most effective ways to apply technology to the prototyping process, a deeper, behavioral model of prototyping must be used. In an attempt to build such a model for the development of large software products, Curtis et al. conducted a field study of large software development projects [24, 25]. They found that behavioral processes in such projects can be analyzed from cognitive, social, and organizational perspectives. Figure 2 updates the model to reflect these three perspectives. In the next three sections, the prototyping process will be examined from each of these perspectives.

2.1 The Cognitive Perspective of the Prototyping Process

The talents, skills, and experience levels of the individual participants of a prototyping project are important influences on the prototyping process. For example, Blum [8] states that the form that the prototyping process takes can be greatly influenced by the experience level of the system builder, while Naumann and Jenkins [65] state that the set of critical skills and abilities required for the system builder using prototyping are significantly different from those required for traditional systems development.

During prototyping, the system builder constructs successive versions of a prototype system, resolving conflicts with the user (requirements) within

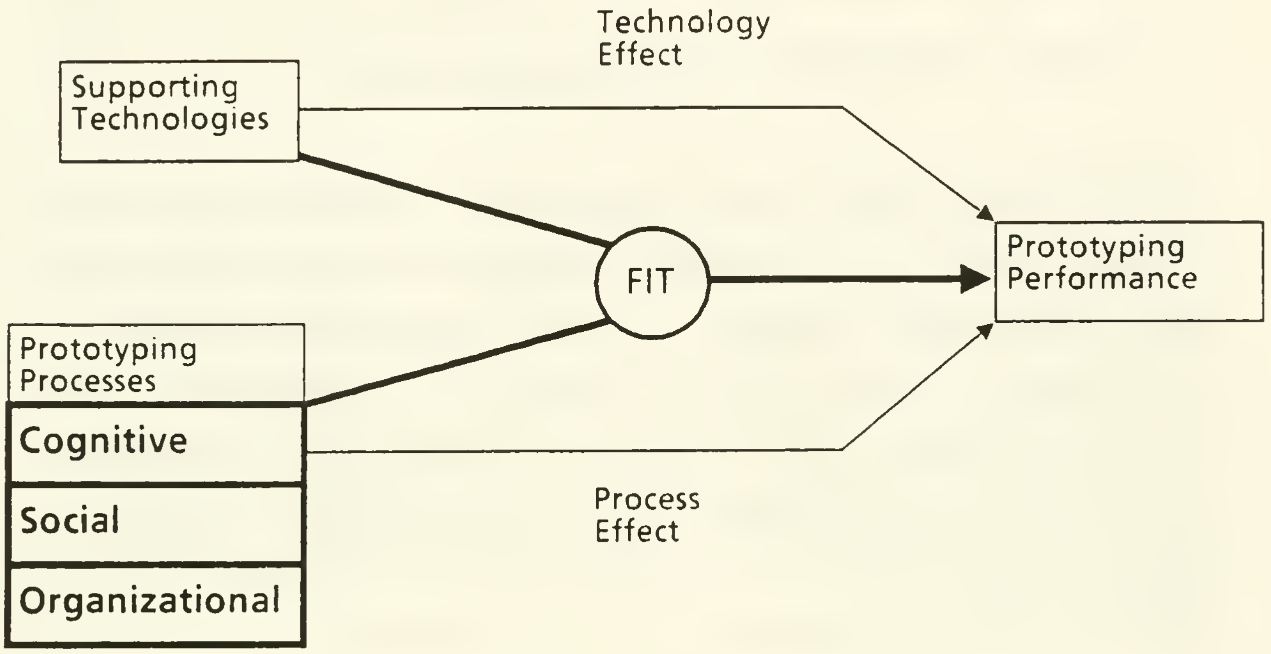

Figure 2 -- The Impact of Support Technology on Prototyping Performance: The Perspectives of the Prototyping Process

technology and economic constraints for each successive version [65]. The builder performs detailed design and program generation as he builds and modifies each version of the system prototype. From a cognitive or individual perspective, the prototyping process primarily focuses on producing new prototype versions as quickly as possible. This perspective is common in the prototyping literature. However, the builder must respond to changing perceptions of need by working closely with the user at each step of the process. This builder-user interaction has been discussed by a number of researchers in the field, and it serves as the foundation for the process perspective discussed in the next section: the social perspective.

2.2 The Social Perspective of the Prototyping Process

There has been a fairly large amount of research on the impacts of social processes on I/S development (see [37] and [38] for discussions of this research area). This research has provided a number of insights into the performance impacts of group behavior on software development. Generally, however, it has been poorly grounded in theory and has not provided models for understanding the performance effects of a wide range of possible changes in group process behavior [50].

Collaboration between the system builder and user is at the heart of prototyping [84]. Prototyping stresses the interactions between the user, builder, and system [65]. Additionally, the manager role is a significant one in prototyping [3, 77] (as it is with software development in general [37]), though it has been generally neglected in the prototyping literature [54]. The relationships between builder, user and manager highlight the fact that prototyping is a social process at its core. Almost by definition, a system builder can not prototype alone. The essence of prototyping is in the dialogue between builder and user [84], within the resource constraints established by management [65].

It is very important for all three roles (builder, user, and manager) to participate fully in the prototyping process. The significance of the manager role was mentioned above. Henderson's empirical results [38] stress the importance of both the user and builder roles in the design process. He found that neither user-dominated nor builder-dominated teams are as high-performing as teams with more balanced levels of influence. Both the user and builder have expertise that is useful in the prototyping process. Such research suggests that the Manager-User-Builder relationships are critical to prototyping processes.

In addition to communication within the team (Manager-User-Builder), communications with non-team stakeholders is an important process for the successful prototyping team. Gladstein found such boundary management to be a major determinant of group performance [33]. Henderson relates this to design teams, arguing that boundary management is required for a team to establish design validity [37], and Zmud found that the innovativeness of a software development group is facilitated if appropriate channels are provided to link the design with relevant external information sources [85]. Both inter- and extra- team communications are important social processes for the successful prototyping team.

An idealized model of the social prototyping process would begin with each member of the team holding his own mental model of the proposed system. The differences and conflicts in the various individuals' mental models would be resolved and compromised until a final prototype product was reached [24, 25]. In this idealized model, the conflicts would be resolved objectively, and the "best" design alternative would eventually be reached.

This ideal model rarely matches the reality of actual prototyping projects. There is a danger in prototyping that the system builder and user will come to agreement on a design solution before they have fully evaluated all of the system design requirements. Because of this, they may converge on a solution that is less than optimal. Mathiassen states that a fast-converging prototyping process may lead to "tunnel vision" -- and to the ignoring of important requirements for the system product [63]. Henderson and Ingraham [40] found that "a comparison with the information requirements generated by a structured group process indicated that prototyping is a convergent design method that may overlook important user information needs." Thus, techniques to balance the rapid convergence of prototyping approaches may have significant performance impacts.

A final social issue involved in prototyping is overall project management. Scharer states that the challenge of managing a prototyping project can be greater than that of managing a project undertaken with a conventional methodology [77]. In fact, most of the gains in productivity and quality that prototyping offers can be realized only if the process is skillfully managed. Alavi found that while prototyping provides clearer and better communication between system developers and users during the design phase of a project, the management and control of the project is more difficult [3].

The successful prototyping project, from a social perspective, effectively utilizes communication between members of the team as well as with non-team stakeholders. Additionally, it requires the examination of alternative designs and concepts to slow the rapid convergence of ideas. Finally, it requires strict project management control. Together, these factors can greatly enhance the social processes inherent in prototyping. It must always be remembered, however, that these processes take place within an organizational environment. This organizational perspective provides the focus for the next section.

2.3 The Organizational Perspective of the Prototyping Process

The prototyping of an information system is performed within an organizational context (as is any software development method). Curtis et al. cite "requirements volatility" as the fundamental organizational issue for software development, feeling that system requirements usually change because of environmental changes facing the organization [25]. Certainly, reacting to a changing environment can be important for the system prototyper. Quick response to such changes has, in fact, been cited as a primary advantage of prototyping [81]. There are, however, other organizational issues that need to be addressed as well.

Bourke describes a prototyping project that was a technical success but was canceled because of inadequate sponsorship by the organization and the lack of what he refers to as a "prototyping infrastructure" [14]. His prototyping infrastructure consists of three factors:

1. A long-range systems plan.
2. A steering committee at the executive level
3. Project funding at the corporate level.

Bourke feels a prototyping project is doomed to fail without this organizational support.

Nosek suggests that decision makers can choose organization structure, reward systems, information processes, and people (using Galbraith's [29] organization design variables) to ensure the successful completion of prototyping projects [68]. He provides examples of two different organizational approaches to prototyping. One organization took an ad hoc approach to prototype development with minimal changes to organization design, while the second demonstrated a full commitment to providing a fertile environment for building prototypes by making substantial changes in organization design. For example, the second organization created a resource center to provide developers with information about the use of prototyping tools.

The organization environment, then, can have a great effect on the prototyping process. The organizational perspective on the prototyping process consists of two general processes. The first involves the coordination of multiple teams. The organization needs to have the ability to create and manage a variety of design teams (perhaps some using prototyping and some not) without a significant reduction in the performance of any single team. The key issues relating to this coordination are the overall control of the various teams and the management of the distributed knowledge inherent in them. The project control and knowledge management can each be maintained on either a centralized or distributed basis, and the choice will greatly impact the technology used to support the project and the project performance.

The second process from an organizational perspective involves learning about and reacting to the external environment (competitors, regulations, etc.). This process is necessary for identifying important applications and keeping applications flexible. The ability to create prototyping products or components that can be leveraged to quickly meet competitive demands can be extremely important for an organization struggling to maintain or improve its competitive position in a turbulent environment.

The organizational perspective of the prototyping process provides a business-level view of the project. It requires intra-company coordination of teams and extra-company gathering of information. This perspective can be enhanced by adjusting the organization design variables and by providing functionality to carry out the intra- and extra- company communication needed.

Having examined the process of prototyping from cognitive, social, and organizational perspectives, we will now characterize the support technology that can be used to aid the system prototyper. This characterization can then be used to describe the possible technology-process fit necessary for performance impact.

3.0 Dimensions of Support Technology

As was discussed in Section 1, the availability of support technology is widely held to be a primary determinant of high-performance for prototyping. The supporting technologies probably mentioned most frequently in the literature are very high-level languages (also referred to as 4th-generation languages, end-user languages, etc.) [20, 26, 65]. However, such technology by itself is not enough to establish a successful prototyping environment. As Taylor and Standish state: "we must have a multi-faceted technical approach to rapid prototyping if we are to address a broad range of prototyping applications successfully [81]." We maintain that the technology must not only be "multi-faceted" but must also "fit" the prototyping processes in use.

Henderson and Cooprider present an empirically-derived functional model of I/S planning and design technology [39]. They list 98 specific technology functions identified by experts in planning and design technology. These 98 functions are categorized into three general dimensions and seven specific technology components. These dimensions are shown to be useful for characterizing the capabilities of commercially available planning and design technologies. They are also useful for characterizing the types of support technologies that are most important in a prototyping environment. The dimensions and their components are listed in Table 1.

1. PRODUCTION TECHNOLOGY
   A. Representation
   B. Analysis
   C. Transformation

2. COORDINATION TECHNOLOGY
   A. Control
   B. Cooperative Functionality

3 ORGANIZATIONAL TECHNOLOGY
   A. Learning/Support
   B. Infrastructure

Table 1 -- The Dimensions of I/S Planning and Design Technology

Table 1 shows three general dimensions for I/S planning and design technology: production, coordination and organizational. *Production* technology has a direct impact on the capacity of an individual to generate planning and design decisions and subsequent artifacts or products.. *Coordination* technology enables or supports the interactions of multiple agents in the execution of a planning or design task. *Organizational* technologies are the functions and associated procedures that determine the environment in which production and coordination technology will be applied to the planning and design process. Figure 3 updates the model to reflect these three dimensions of technology.

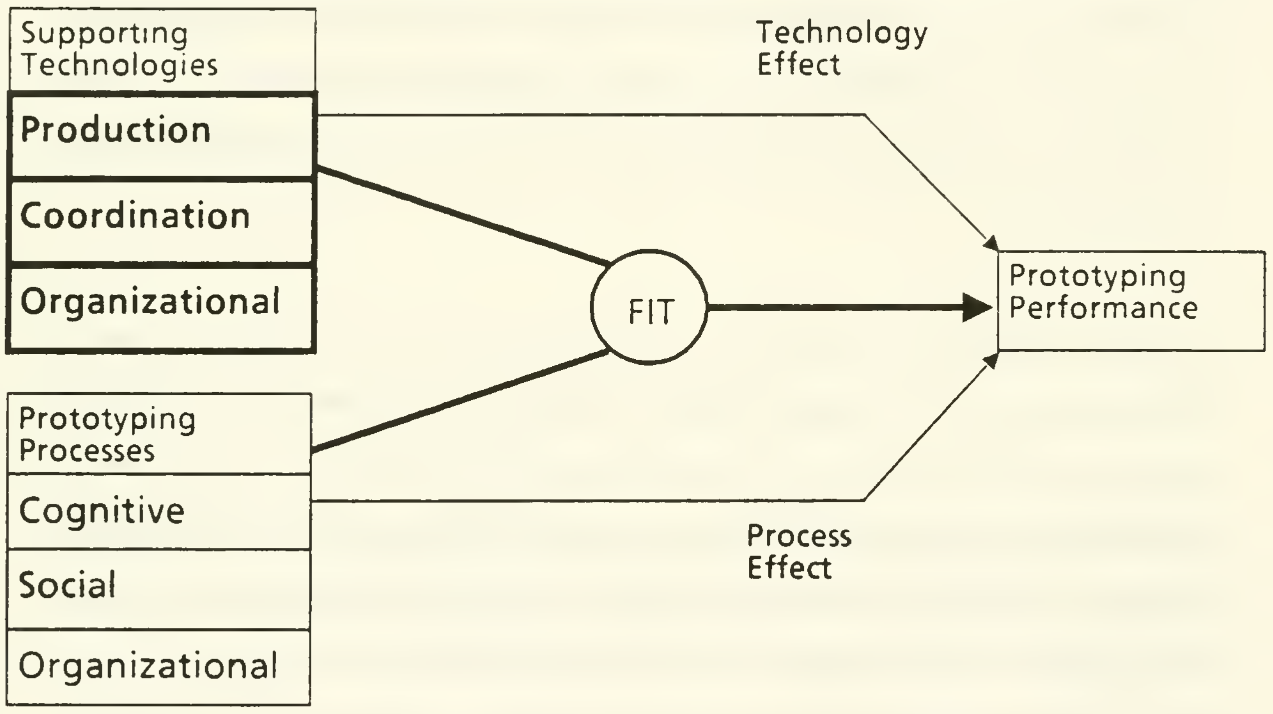

Figure 3 -- The Impact of Support Technology on Prototyping Performance: The Dimensions of Supporting Technology

3.1 The Production Dimension of Technology

Within each of the general technology dimensions, more specific components characterize the technology. Production technology consists of three components: *representation*, *analysis* and *transformation*. *Representation* technology enables the user to define, describe or change a definition or description of an object, relationship, or process. Such capabilities are are at the heart of traditional characterizations of prototyping technology. By its nature, prototyping is a representation or modeling process [73]. Naumann and Jenkins explicitly list "modeling" as a required technology for prototyping [65]. Any technology supporting prototyping must provide an array of representation functionality.

Analysis technology enables the user to explore, simulate, or evaluate alternate representations or models of objects, relationships or processes. It is very similar to Riddle's "technology of evaluation" [73]. As was discussed in Section 2.2, many authors have specifically criticized prototyping as a convergent design process. Analysis technology provides functionality for encouraging the examination of design alternatives and implications .

Transformation technology executes a significant prototyping task, thereby replacing or substituting for a human system builder. This dimension reflects a straight capital/labor substitution: technology for human labor. This functionality has received a great deal of attention in the prototyping literature. Prototyping implies a need to do things quickly (e.g. "rapid" prototyping), and transformation technology can play a major role in producing a prototype quickly. For example, Feather describes a system in which a formal specification is automatically "transformed" into an implementation -- a prototype [27].

The transformation functionalities currently available in support technologies in the marketplace are generally focused relatively late in the software development life cycle [39]. For example, code generation is a typical transformation technology, and it is frequently used only after much of the design work is completed. In contrast, prototyping often provides much of its value *early* in the design life-cycle, during problem formulation and definition [12]. The determination of the precise types of transformation technology that provide the most impact on prototyping processes is an important subject for further research.

3.2 The Coordination Dimension of Technology

The need for coordination stems from the resources that are expended in tasks that require multiple actors or agents [59]. The general dimension of coordination technology consists of two more specific components: *control* and *cooperative functionality*. *Control* technology enables the user to plan for and enforce rules, policies or priorities that will govern or restrict the activities of team members during the prototyping process. There are two types of control covered in this dimension: access control and resource management. Access control assumes that issues of security and access must be carefully managed. Resource management enables a manager to plan for, allocate and monitor the use of design team resources.

Two control issues that are especially important for prototyping deserve special mention. As discussed in Section 2.2, project management can be extremely difficult in a prototyping environment. The familiar check-points inherent in the traditional development life cycle are generally lacking in prototyping [2], and it can be difficult for managers to monitor and control team members in a prototyping environment. Project control functionality is thus critical -- probably even more critical than for a traditionally managed development project. In addition to project control, general software configuration management functions [6, 67] or specific version control functions [41] can be very important. Since prototyping is an iterative process -- with a new prototype system being generated at each iteration -- control tools such as these are very useful for tracking and managing the process.

Cooperative functionality enables the user to exchange information with other individuals for the purpose of influencing (affecting) the concept, process or product of the prototyping team. This dimension of support technology has been largely ignored in the prototyping literature. However, if one accepts the social perspective of prototyping the importance of cooperative functionality becomes evident. The ability to exchange information between members of the team and between the team and outsiders can be a significant determinant of performance. The ability of the coordination technology to function both as a communication channel and as a facilitation aid can be very important for the prototyping system builder. Examples of such technology can be found in research on group DSS [44, 55] and computer-supported cooperative work (see the *Proceedings of the Conference on Computer-Supported Cooperative Work*, Austin, Texas, December 3-5, 1986). Plexsys, for example, is a system which provides support for group brainstorming and evaluation that can both increase the efficiency of group meetings and counter the rapid convergence seen in many prototyping projects [5].

3.3 The Organizational Dimension of Technology

Finally, the organizational dimension consists of two specific components: *learning/support* and *infrastructure*. *Learning/support* technology helps an individual user understand and effectively use the available technology. Such functionality helps the system builder become productive more quickly in new environments by making it easier to learn the available tools. Additionally, it can improve the Builder-User relationship by allowing users to use available technologies to explore their own alternatives -- potentially making users a more influential part of the design team. It also aids the organization by providing training to a minimum skill level for all members across all teams -- assisting communication and cooperation between them.

Infrastructure technology is defined as functionality standards that enable portability of skills, knowledge, procedures, or methods across planning or design processes. Infrastructure technology gives teams the ability to share models, code, documentation, etc. By allowing teams to communicate and coordinate through the use of standards, the performance of the entire I/S organization (as well as that of individual teams) can be markedly improved. Hyer and Wemmerlöv describe how standards for parts codes can save manufacturing organizations great amounts of time and money [47]. In the I/S area, standards as performance enhancers are not as well understood. Jeffery, for example, states that programming and documentation standards have been suggested as affecting I/S development performance, but their specific productivity impacts have not been investigated [50].

3.4 Technology Summary

Most of the support technology that has been suggested in the literature for prototyping environments is production technology. Such technology enables the individual system builder to build and analyze prototype systems as rapidly as possible. However, this technology will only marginally increase the performance of I/S development teams and organizations as long as it is used to just "do the same things faster". If the production technology enables the system builders to "do different things", then the performance can increase by a much larger amount. Order of magnitude improvement can only be achieved when the technology changes the process of software development [50]. For example, code generation is a typical transformation technology [39], and it is frequently suggested as a valuable prototyping tool [20, 73]. However, "coding" typically accounts for only 15 percent of the software costs of large systems [56]. Using code generation to double coding efficiency, therefore, will provide only a 7.5 percent productivity increase. However, if code generation is used as a way to generate successive prototype systems more quickly and this leads to better evaluation of the system design, then design defects can be significantly reduced [52] and more accurate systems requirements can be established early in the development life cycle, greatly reducing total project costs [53].

Gains in overall performance can be further leveraged by the use of technologies that support the social and organizational perspectives of the prototyping process. Support technologies improve performance by providing not only the "individual-based" representation-analysis-transformation functionality of production technology but also by aiding intra-team coordination and boundary management through the use of control and cooperative support functionalities. Additionally, support technology can help in the establishment of an organizational infrastructure for inter-team coordination by providing learning/support functions and technology standards for the entire organization.

It is expected that direct performance impacts will result from applying production technology to prototyping processes in the cognitive perspective, coordination technology to the social perspective processes, and organizational technology to the organizational perspective processes. If, however, individuals use production technology in ways that allow them to function more effectively as a team, there may be a much larger performance impact. Similarly, coordination technology will directly impact the social processes of prototyping, but it may have a larger effect on performance if it is used by the organization to leverage its management of the entire development process. For example, such technology may allow the parallel operation of multiple teams by providing a common technology infrastructure. Thus, direct performance impacts generally occur across corresponding levels (production-cognitive, coordination-social, organizational-organization), but impacts across levels may enable much greater impacts on prototyping performance.

4.0 Measuring the Performance of Prototyping Projects

Performance measurement is a central issue for evaluating the impact of support technology on the prototyping process. Churchman has stated that all design is goal-directed and performance-based [22]. In effect, design is not possible without a measure of performance to support it. Hirschheim and Smithson describe performance measurement as the link between evaluation and tools [42]. To establish this link, appropriate measures must be defined. As Strassman states: "You cannot measure what is not defined. You also cannot tell whether you have improved something if you have not measured its performance ([80], p. 100)." In attempting to describe the impact of particular technologies on the prototyping process, performance measurement is especially critical.

Research attempting to evaluate I/S performance has suggested a range of possible measures. Hirschheim and Smithson, however, state that I/S evaluations have been misdirected toward tools and techniques for measurement and away from understanding [42]. For example, many researchers have based their productivity measurements on the number of lines of code produced by a given amount of effort [56]. This approach has long been recognized to have a number of obvious problems -- each code line does not have the same value and does not require the same amount of time or effort to produce, for example. However, most software productivity researchers today still use lines of code as a primary focus [13], though function point measurement has recently gained a level of popularity [4]. Performance measures such as "source lines of codé per month" are relatively easy to gather and use. However, focusing too closely on them may distort the overall productivity picture and cause managers to overlook promising avenues for performance improvements. The ultimate goal, of course, is to provide applications that benefit the organization and not to simply create lines of code. Measures like lines of code do not address the issues of concern to management, particularly senior management. A more comprehensive view of I/S performance is required. Figure 4 illustrates such a view of performance.

	Process	Product
Task Measures	Efficiency	Quality
Social Measures	Synergy	Validity
Strategic Measures	Flexibility	Leverage

Figure 4 -- A Framework for Measuring I/S Prototyping Performance

As seen in Figure 4, one major issue in measuring I/S performance is reflecting both product and process dimensions. Many authors have suggested the value of separate I/S measures for the products and processes of I/S development. Agresti used an industrial engineering perspective to describe process and product measures of software development [1]. Riddle [73] and Sol [78] have discussed the value of this general approach in a prototyping environment. Since the ultimate goal of performance measurement is to improve the processes involved, these authors state, not only must the final product of the prototyping effort be evaluated, but the process used to obtain that product must also be measured. It is important to use *both* types of measures because there is a potential conflict between the efficiency of the process and the quality of the product [28].

There can be difficulties in applying this process/product perspective to prototyping, however. The "product" of the prototyping process is sometimes not available at the end of a development project. Taking their cue from Brooks' [15] admonition to "plan to throw one away," many prototypers [31, 63, 71, 73, 75] feel that once a prototype has served its purpose of defining user requirements, it can (and should) be "disposed of". The debate of "expendable" vs. "evolutionary" prototyping [45] has received a great deal of attention [8, 35, 61]. For our purposes, this issue is somewhat moot. The critical issue is to include the effort to produce a final product as only one of several measures for evaluating performance.

The other dimension of performance shown in Figure 4 illustrates that different behavioral perspectives lead to differing measurements. This reflects the view that the measurement of prototyping performance should be oriented toward management decision and action. Mason and Swanson have recommended an approach toward measurement for management that is very different from traditional "scientific" measurement -- adding concern for behavioral dimensions, for example [60]. Boehm clearly shows that the perceptions of upper management, middle management, and programmers are very different concerning the factors that have the most leverage for affecting development productivity [11]. These perceptions can be important motivational factors for affecting the prototyping process, and they should be reflected in any performance measurement system. Buss found that top executives, I/S managers and users often have conflicting views of which project proposals should get approval [18]. These conflicting views can be mitigated to some degree if the language (or measures, in this context) used by each perspective are the same, or are at least translatable.

The performance measurement system presented in Figure 4 uses the perspectives of task, social, and strategic to reflect this dimension of measurement. These three perspectives generally match the process perspectives that were developed in Section 2, and they match well with the technology impacts on those processes. The following three sections describe each of the perspectives of performance measurement.

4.1 Task Measures of Performance

The task measures of Figure 4 are indicators of the accomplishment of the builder's defined tasks. They are "individual oriented" and are concerned with "doing things right". *Task process* measures are primarily used to evaluate the efficiency of individual programmers. Basically, they represent typical measures of the economic efficiency of production. Source lines of code or function points per month are typical measures of this perspective. *Task product* measures evaluate the quality of the program product. These measures typically take one of two forms. First, quality is measured in terms of defects per unit output. For software development, this is expressed in terms of bugs per thousand lines of code, for example. The second form of quality measurement evaluates the product's overall technical performance rather than simply counting defects. Examples of this form of task product measurement include run-time execution speed and object code size.

These task-oriented measures dominate the I/S performance literature [42]. There are a number of problems, however, with focusing exclusively on these types of measures. Some of the specific complaints about these types of measures were described earlier. In general, they are measures of output (in an economic sense) rather than outcome (the total impact of the process) [70]. They are much more closely aligned with efficiency (how well input is converted to output) than with effectiveness (how the input is used to accomplish the goals of the organization). The addition of social and strategic perspectives provides a more complete performance evaluation.

4.2 Social Measures of Performance

Hirschheim and Smithson argue that both technical (task-oriented) and non-technical (social) criteria must be included for I/S evaluation to be meaningful [42]. The social measures of this dimension evaluate how well the prototyping team performs as a whole. They are "group oriented" and are concerned with "doing the right things". The *social process* measures test how well the development team functions together. We refer to these types of measures as "synergy" measures. They evaluate the process gains or losses that occur because prototyping project members work as a group rather than as individuals. In terms of group theory, this is mainly an issue of team maintenance [33], and involve such things as commitment, quality of work life, and satisfaction with the team. Steiner describes the impact of group process on performance in more traditional economic terms [79]. He characterizes the performance of groups as the sum of the performances of the individual members of the team with the group process gains or losses. An interesting measure of group process in this light is the prototyping team "efficiency" (defined as it was above for individuals: function points per month, for example) divided by the sum of the efficiencies of the individual team members. An alternative measure of this dimension is each individual's efficiency on the current prototyping project divided by his "historical" efficiency -- his efficiency on past projects and teams.

The *social product* measures evaluate design validity -- is the team meeting the project requirements and do the requirements satisfy a real business need? These measures can be critical in providing what Bjorn-Andersen [7] refers to as a "challenge to certainty." He states that I/S developers tend to spend more and more time and use ever more refined technological tools for solving the wrong problem more precisely. Support technology must provide the ability to test design validity, and the best source of these tests comes from communication with non-team stakeholders [22, 37]. The performance measures for this dimension include user satisfaction and the rate of function change over time (increasingly smaller amounts of change in the system -- given the same system usage -- imply a more accurate design solution). Similarly, system usage and the link between that usage and the behavior of the organization are sources of validity measurements. In essence, a valid design is one that affects an intended change in organizational behavior.

The problem of performance measurement occurs within an organizational context, and it is important that such measurement be used with an organizational perspective. This is an example of what Churchman refers to as the "systemic" level of measurement: placing a manager's problem situation in a larger context [21]. The next section looks at the manager's problem of performance measurement from a larger organizational perspective.

4.3 Strategic Measures of Performance

Very little progress has been made in discovering organizational measures of performance [23, 70]. The measures of performance in this dimension involve business success. At the core of these measures is a concern with using technology to innovate -- with "doing things differently". *Strategic process* measures are primarily concerned with organizational flexibility. The motivation for this view is the need to manage organizational resources in the face of significant environmental changes. There are two general perspectives for these measures. The first perspective is one of traditional project management. Measures in this perspective evaluate the project's ability to stay on schedule and within budget. The second perspective becomes more important as the organizational environment becomes increasingly turbulent. It focuses on the organizational resource of *time*. The main concern in this perspective is the speed with which the organization reacts to changes in the environment. A fundamental measure in this area is time-to-break-even: the length of time it takes an organization to convert an I/S concept into a viable product that has earned back its development costs for the organization. This measure highlights the criticality of rapid response to environmental changes while directly incorporating aspects of quality and maintainability. It is rapidly becoming a key measure of I/S performance [9].

The *strategic product* issue is fundamentally one of business leverage -- the use of prototyping to leverage a position of competitive advantage for the organization. It also reflects the importance of continually trying to do new things in a changing environment. As Sir Francis Bacon said in his essay "Of Innovations": "He that will not apply new remedies must expect new evils; for time is the greatest innovator" (as quoted in [49]). Time will leave behind those organizations whose I/S leaders are not willing to innovate -- transforming the business. For example, if systems were built in ways to facilitate future reuse of code, prototypers could leverage their development efforts by using existing (debugged) code modules. In addition, the final product of their efforts would result in more code modules that could be leveraged for future products. The ultimate measures in this area primarily come from competitive analysis and environmental scanning: market share over time, for example, or rate of new product introductions compared to the rates of competitors. However, the percentage of reused code in a project is a possible intermediate indicator.

4.4 Performance Impacts

Figure 5 shows the completed framework. As suggested in previous sections, the major impact of production technology is on the cognitive processes of prototyping, and its direct performance impact will primarily be on the task measures of performance. For example, transformation technology such as code generation are expected to increase programmer efficiency, and analysis technology such as consistency tests between alternate representations will improve the quality of the program product. Coordination technology's direct impact will be on social processes as well as the social measures of performance. For example, control technology can be used to implement flexible project management strategies, giving team members more autonomy and increasing their job satisfaction [34]. Additionally, cooperative functionality such as dialogue management can be used to involve non-team stakeholders in the prototyping process, thus increasing design validity [37]. Organizational technology will have its most direct impact on organizational processes and on strategic measures of performance. For example, the use of standards provides common communication protocols and knowledge representations, allowing the time-sharing of knowledgeable members among teams. This gives more teams better access to project knowledge, leading to a more effective use of the organization's knowledge resources and more flexible prototyping processes. Additionally, learning/support technology such as libraries of reusable code modules can help prototypers leverage their efforts to produce new products more quickly -- enabling their organizations to better attain competitive advantage.

As was discussed in Section 3.4, however, some of the largest effects on productivity -- the order-of-magnitude changes -- come from the ability to change the existing prototyping processes. For example, the potential for technology to empower a given role can radically change the group dynamics in a design process. Representation and analysis technology can reduce the design

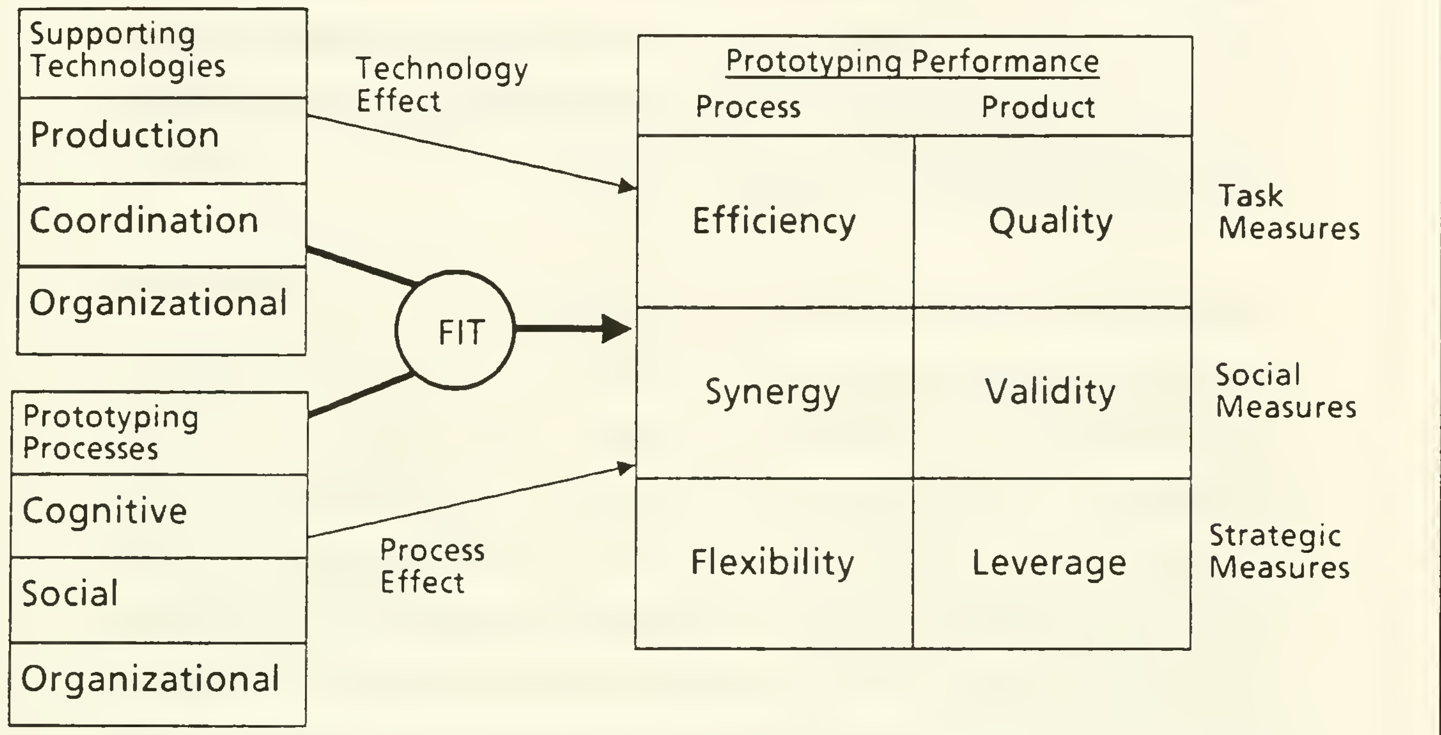

Figure 5 -- The Impact of Support Technology on Prototyping Performance

knowledge required for users to effectively participate and as a result the user role becomes much more active in the prototyping process.

5. Conclusions and Recommendations for Future Research

This paper has examined the performance impacts of support technology on the prototyping process. In doing this, the prototyping process was examined from three behavioral perspectives: cognitive, social, and organizational. Support technology was characterized by the functional dimensions of production, coordination and organizational technology. A performance measurement framework was developed using the dimensions of process/product and task/social/strategic perspectives. This framework was used to base discussions of examples of performance impacts.

Much research remains to be done in this area in the future. Development of performance measures for each of the dimensions through the use of applicable reference disciplines is an obvious first step. Just as Agresti used industrial engineering as a reference field for measuring I/S productivity [1], one could envision using research in manufacturing, group processes, economics, organizational studies, etc., to derive new measurements for prototyping performance. For example, a prime field to examine might be the impact of group technology on manufacturing productivity -- an area in which a number of studies have already been made [46, 30].

Perhaps the major research issue emerging from this paper is the criticality of the technology-process "fit" to prototyping performance. Will a technology platform enable major performance impacts by itself or must the technology also match the specific design methods used? In other words, should design processes be altered to match the technology in use (through training, etc.) or should technology be modified to match design processes? Would the cost of either alternative be justified by the performance impacts they would provide? In the final analysis, the question of the magnitude of the impact of the technology-process "fit" as compared to the impacts of either the process or technology effects alone is an empirical issue, and it must be tested as such in the future.

It should be noted that many of the concepts presented in this paper are not limited to the prototyping environment. Though the process-technology-performance issues discussed were directly framed for prototyping, the implications of the process-technology "fit" and the performance measurement framework should be generally applicable to a wide range of design methodologies. Applications to other methods should be made in the future.

We have presented perspectives on the measurement of prototyping project performance, and a way to assess technologies that are available for supporting the prototyping environment. This paper highlights the importance of examining the performance impacts of support technology in a broad manner, and suggests that such an examination can be done in a systematic manner. Of course, the ability to attribute performance impacts to support technology becomes increasingly difficult as one moves from an individual unit of analysis to an organizational one. However, the ability to characterize prototyping behaviors and support technologies along the presented dimensions suggests that these frameworks can lead to an increased level of understanding of performance determinants at all levels..

References

1. Agresti, W. "Applying Industrial Engineering to the Software Development Process," in *Proceedings: IEEE Fall COMPCON*, Washington D.C.: IEEE Computer Society Press, 1981, 264-270.

2. Agresti, W. (ed.) *New Paradigms for Software Development.* Washington, D.C.: IEEE Computer Society Press, 1986.

3. Alavi, M. "An Assessment of the Prototyping Approach to Information Systems Development." *Communications of the ACM.* Vol. 27, No. 4. 1984, 556-563.

4. Albrecht, A. "Measuring Application Development Productivity," *Proceedings of the Joint SHARE/GUIDE Symposium*, October 1979, 83-92.

5. Applegate, L., B. Konsynski and J. Nunamaker. "A Group Decision Support System for Idea Generation and Issue Analysis in Organization Planning," in *Proceedings of the Conference on Computer-Supported Cooperative Work*, Austin, Texas, December 3-5, 1986, 16-34.

6. Babich, W. *Software Configuration Management: Coordination for Team Productivity.* Reading: Addison-Wesley, 1986.

7. Bjorn-Andersen, N. "Challenge to Certainty," in T. Bemelmans-(ed.) *Beyond Productivity: Information Systems Development for Organizational Effectiveness.* Amsterdam: North-Holland, 1984.

8. Blum, B. "Application Systems Prototyping" in *Prototyping: State of the Art Report.* Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 3-14.

9. Boddie, J. *Crunch Mode: Building Effective Systems on a Tight Schedule*, Englewood Cliffs, NJ: Prentice-Hall, 1987.

10. Boehm, B. "Software Engineering," *IEEE Transactions on Computers*, Vol C-25, No. 12, December 1976, 1226-1241.

11. Boehm, B. "Improving Software Productivity," in *Proceedings: IEEE Fall COMPCON*, Washington D.C.: IEEE Computer Society Press, 1981, 264-270.

12. Boehm, B. "Verifying and Validating Software Requirements and Design Specifications", *IEEE Software*, January 1984, 75-88.

13. Boehm, B. "Improving Software Productivity," *Computer*, Vol. 20, No. 9, September 1987, 43-57.

14. Bourke, M. "Actual Experiences in Prototyping," in *Prototyping: State of the Art Report.* Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 15-26.

15. Brooks, F. *The Mythical Man-Month: Essays on Software Engineering,* Reading, MA: Addison-Wesley, 1975.

16. Brooks, F. "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer*, April 1987, 10-19.

17. Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). *Approaches to Prototyping.* Berlin: Springer-Verlag, 1984.

18. Buss, M. "How to Rank Computer Projects," in R. Galliers (ed.) *Information Analysis: Selected Readings.* Sydney: Addison-Wesley, 1987, 395-407.

19. Card, D., F. McGarry, and G. Page. "Evaluating Software Engineering Technologies," *IEEE Transactions on Software Engineering*, Vol SE-13, No. 7, July 1987, 845-851.

20. Carey, T. and R.E.A. Mason. "Information System Prototyping: Techniques, Tools, and Methodologies," in *New Paradigms for Software Development.* W. Agresti (ed.), Washington, D.C.: IEEE Computer Society Press, 1986, 48-58.

21. Churchman, C. "Suggestive, Predictive, Decisive, and Systemic Measurement." Paper presented at the 2nd Symposium on Industrial Safety Performance Measurement, National Safety Council, Chicago, Dec. 1968.

22. Churchman, C. *The Design of Inquiring Systems.* Basic Books, 1971.

23. Crowston, K. and M. Treacy. "Assessing the Impact of Information Technology on Enterprise Level Performance," in *Proceedings of the Seventh International Conference on Information Systems*, 1986.

24. Curtis, B., H. Krasner, V. Shen, and N. Iscoe. "On Building Software Process Models Under the Lamppost," in *Proceedings of the 9th International Conference on Software Engineering.* IEEE Computer Society Press, 1987, 96-103.

25. Curtis, B., H. Krasner, and N. Iscoe. "A Field Study of the Software Design Process for Large Systems," *Communications of the ACM.* Vol. 31, No. 11, 1988, 1268-1286.

26. Evans, F. "Selling Prototyping Techniques to Management," in *Prototyping: State of the Art Report.* Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 27-40.

27. Feather, M. "Mappings for Rapid Prototyping," *ACM SIGSOFT Software Engineering Notes: Special Issue on Rapid Prototyping*, Vol. 7, No. 5, December 1982.

28. Floyd, C. "A Systematic Look at Prototyping," in *Approaches to Prototyping*. Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 1-18.

29. Galbraith, J. *Organization Design*. Reading, MA: Addison-Wesley 1977.

30. Gallagher, C. and W. Knight. *Group Technology Production Methods in Manufacture*. West Sussex: Ellis Horwood. 1986.

31. Gehani, N. "A Study in Prototyping," *ACM SIGSOFT Software Engineering Notes: Special Issue on Rapid Prototyping*, Vol. 7, No. 5, December 1982.

32. Gladden, G. "Stop the Life Cycle I Want to Get Off," *ACM SIGSOFT Software Engineering Notes*. Vol. 7, 1982.

33. Gladstein, D. "Groups in Context: A Model of Task Group Effectiveness," *Administrative Science Quarterly*, Vol. 29, 1984.

34. Goldstein, D. and J. Rockart. "An Examination of Work-Related Correlates of Job Satisfaction in Programmer/Analysts," *MIS Quarterly*, Vol. 8, No. 2, June 1982, 103-116.

35. Gomaa, H. "Prototypes -- Keep Them or Throw Them Away?" in *Prototyping: State of the Art Report*. Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 41-54.

36. Hackathorn, R. and J. Karimi. "Comparative Evaluation of Information Engineering Methods", Informations Systems Research Group, College of Business and Administration, University of Colorado at Denver, February 1986.

37. Henderson, J. "Managing the IS Design Environment." Center for Information Systems Research Working Paper No. 158, Sloan School of Management, M.I.T., Cambridge, MA, May 1987.

38. Henderson, J. "Involvement as a Predictor of Performance in I/S Planning and Design." Center for Information Systems Research Working Paper No. 175, Sloan School of Management, M.I.T., Cambridge, MA, July 1988.

39. Henderson, J. and J. Cooprider. "Dimensions of I/S Planning and Design Technology". Center for Information Systems Research Working Paper No. 181, Sloan School of Management, M.I.T., Cambridge, MA, September 1988.

40. Henderson, J. and R. Ingraham. "Prototyping for DSS: A Critical Appraisal," in Ginzberg, M., W. Reitmann, and E. Stohr (eds.) *Decision Support Systems*. Amsterdam: North Holland 1982, 79-96.

41. Henderson, J. and D. Schilling. "Design and Implementation of Decision Support Systems in the Public Sector," *MIS Quarterly*, Vol. 9, No. 2, June 1985, 157-170.

42. Hirschheim, R. and S. Smithson. "Information Systems Evaluation: Myth and Reality," in R. Galliers (ed.) *Information Analysis: Selected Readings.* Sydney: Addison-Wesley, 1987, 367-380.

43. Huber, G.. "The Nature of Organizational Decision Making and the Design of Decision Support Systems." *MIS Quarterly*, Vol. 5, No. 2, June 1981, 1-10.

44. Huber, G. "Issues in the Design of Group Decision Support Systems," MIS Quarterly, Vol. 8, No. 3, September 1984, 195-204.

45. Huffaker, D. "Prototyping Business Applications Within the Traditional Life Cycle," *Journal of Information Systems Management*, Vol. 3, No. 4, Fall 1986, 71-74.

46. Hyer, N. (ed.) *Capabilities of Group Technology.* Dearborn: The Computer and Automated Systems Association of SME, 1987.

47. Hyer, N. and U. Wemmerlöv. "Group Technology and Productivity," *Harvard Business Review*, July/August 1984.

48. Iggulden, D. "Prototyping Developments," in *Prototyping: State of the Art Report.* Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 55-64.

49. Izzo, J. *The Embattled Fortress.* San Francisco: Jossey-Bass, 1987.

50. Jeffery, D. "Software Engineering Productivity Models for Management Information System Development," in R. Boland and R. Hirschheim (eds.) *Critical Issues in Information Systems Research.* Chichester: John Wiley & Sons, 1987, 113-134.

51. Jenkins, A. et al. "An Annotated Bibliography on Prototyping." Discussion paper no. 228, 259 (updated annually) Division of Research, Graduate School of Business, Indiana University. April 1984.

52. Jones, T. C. "A Survey of Programming Design and Specification Techniques," in *Proceedings of Conference on Reliable Software*, 1979, 91-103.

53. Jones, T. C. *Programming Productivity.* New York: McGraw-Hill, 1986.

54. Kensing, F. "Property Determination by Prototyping," in *Approaches to Prototyping.* Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 322-340.

55. Kraemer, K. and J. King. "Computer-based Systems for Group Decision Support: Status of Use and Problems in Development," in *Proceedings of the Conference on Computer-Supported Cooperative Work*, Austin, Texas, December 3-5, 1986.

56. Kull, D. "To Raise Productivity Work Smarter, Not Harder," *Computer Decisions*, March 1984, 164-189.

57. Lipp, M. (ed.) *Prototyping: State of the Art Report.* England: Pergamon Infotech Limited, 1986.

58. Lipp, M. "Integration of Prototyping Into a Business Environment," in *Prototyping: State of the Art Report.* Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 145-152.

59. Malone, T. "What is Coordination Theory?" Working Paper #2051-88, Sloan School of Management, M.I.T., Cambridge, MA, 1988.

60. Mason, R. and E. Swanson. "Measurement for Management Decision: A Perspective," *California Management Review*, Vol. 21, No. 3, Spring 1979, 70-81.

61. Mason, R.E.A. "Prototyping in Software Engineering -- Prototyping the Future," in *Prototyping: State of the Art Report.* Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 75-88.

62. Mason, R.E.A. and T. Carey. "Prototyping Interactive Information Systems," *Communications of the ACM*, Vol. 26, No. 5, May 1983, 347-354.

63. Mathiassen, L. "Summary of the Systems Development and Prototyping Working Group," in *Approaches to Prototyping.* Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 255-260.

64. McCracken, D. and M. Jackson. "Life-Cycle Concept Considered Harmful," *ACM SIGSOFT Software Engineering Notes*, Vol. 7, 1982.

65. Naumann, J. and A. Jenkins. "Prototyping: The New Paradigm for Systems Development." *MIS Quarterly*, Vol. 6, No. 3, September 1982, 29-44.

66. Necco, C., C. Gordon, and N. Tsai. "Systems Analysis and Design: Current Practices," *MIS Quarterly*, Vol. 11, No. 4, December 1987, 461-478.

67. Nelson, D. "A Software Development Environment Emphasizing Rapid Prototyping," in *Approaches to Prototyping.* Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 136-151.

68. Nosek, J. "Organization Design Choices to Facilitate Evolutionary Development of Prototype Information Systems," in *Approaches to Prototyping.* Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 341-355.

69. Osterweil, L. "Software Processes are Software Too," in *Proceedings of the 9th International Conference on Software Engineering.* IEEE Computer Society Press, 1987, 96-103.

70. Packer, M. "Measuring the Intangible in Productivity," *Technology Review*, February/March 1983, 48-57.

71. Patton, B. "Prototyping -- A Nomenclature Problem", *ACM SIGSOFT Software Engineering Notes*, Vol. 8, No. 2, April 1983.

72. Pliskin, N. and P. Shoval. "End-User Prototyping: Sophisticated Users Supporting System Development." *DATABASE*, Summer, 1987, 7-12.

73. Riddle, W. "Advancing the State of the Art in Software System Prototyping," in *Approaches to Prototyping*. Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 19-28.

74. Rockart, J. and L. Flannery. "The Management of End User Computing," *Communications of the ACM*, Vol. 26, No. 10, October 1983, 776-784.

75. Rzevski, G. "Prototypes versus Pilot Systems: Strategies for Evolutionary Information System Development," in *Approaches to Prototyping*. Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 356-367.

76. Sarvari, I. "Implementing a Prototype," in *Prototyping: State of the Art Report*. Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 89-103.

77. Scharer, L. "The Prototyping Alternative," in *New Paradigms for Software Development*. W. Agresti (ed.), Washington, D.C.: IEEE Computer Society Press, 1986, 59-68.

78. Sol, H. "Prototyping: A Methodological Assessment," in *Approaches to Prototyping*. Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984, 368-382.

79. Steiner, I. *Group Process and Productivity*. New York: Academic Press, 1972.

80. Strassman, P. *Information Payoff, the Transformation of Work in the Electronic Age*. New York: The Free Press, 1985.

81. Taylor, T. and T. Standish. "Initial Thoughts on Rapid Prototyping Techniques," in *New Paradigms for Software Development*. W. Agresti (ed.), Washington, D.C.: IEEE Computer Society Press, 1986, 38-47.

82. Tavolato, P. and K. Vincena. "A Prototyping Methodology and its Tool," in *Approaches to Prototyping*. Budde, R., K. Kuhlenkamp, L. Mathiassen, H. Zullighoven (eds.). Berlin: Springer-Verlag, 1984.

83. Venkatraman, N. "The Concept of Fit in Strategy Research: Towards Verbal and Statistical Correspondence," Sloan School of Management Working Paper #1830-86, M.I.T., Cambridge, MA, 1986.

84. West, M. "A Taxonomy of Prototyping -- Tools and Methods for Database, Decision Support and Transaction Systems," in *Prototyping: State of the Art Report*. Lipp, M. (ed.) England: Pergamon Infotech Limited, 1986, 105-122.

85. Zmud, R. "The Effectiveness of External Information Channels in Facilitating Innovation Within Software Development Groups," *MIS Quarterly*, Vol. 7, No. 2, 1983, 43-58.