A Prototyping Method for Applications Development by End Users and Information Systems Specialists

By: James M. Kraushaar
Larry E. Shirland
School of Business Administration
Mansfield House
University of Vermont
Burlington, Vermont

Abstract

A prototyping development method is presented here which has the potential to reduce the growing application development backlog. Prior research and our findings indicate that a prototyping process can assist in the efficient development of application systems by breaking a complex problem into several comprehensive parts.

A state-transition model of the IS development process is presented and discussed. A two-prototype method is explained in the context of this model. Two projects are described which are typical of development efforts made by end users in a microcomputer environment and IS specialists in a mainframe environment.

Keywords: Information systems, systems analysis, systems design, methodology, application development.

ACM Categories: D.2.1, D.2.9, K.6.1, K.6.3

Introduction

Many information systems (IS) managers face the challenge of reducing a rapidly growing and seriously underestimated applications development backlog. One strategy for doing this would be to reduce the demand for application systems by raising the price of development. This might be accomplished by developing realistic chargeback systems and/or by creating bureaucratic hurdles. Obviously, this latter approach contains significant risk that the users will satisfy their demand without utilizing the IS department. The potential problems with systems developed without the assistance of the IS department are numerous [3, 17].

Another approach would be to increase the IS department's ability to supply applications development by improving productivity with respect to systems development. This can be accomplished by either developing systems more quickly, or by providing proper development guidelines and assistance so that functional areas can develop their own application systems with minimal IS department resources.

Several methods have been suggested to improve the productivity of the IS department's applications development effort. For example, Martin [13] suggests that the information center approach to IS department organization encourages the supply of proper development guidance and assistance so that functional areas can develop some of their own systems. According to Howdon [9], automated systems development also shows promise in reducing the application backlog; however, it currently requires resources and expertise that many IS departments may not have available.

Prototyping is another approach that can be used to reduce the applications backlog by producing new systems more quickly and effectively than the traditional approach. Unlike automated systems development, prototyping is not sophisticated or expensive. Canning [7] suggests it can be used by both IS department development personnel and end user groups, and Jenkins [10] indicates it is applicable to a wide variety of system development projects.

Our experience, and that of others [6, 8, 18, 19], support these claims. In addition, research find-

ings of Katz [11] in job redesign and Behrens [2] in project development productivity measurement indicate that a prototyping process can encourage the efficient development of application systems by breaking a complex, and often ill-defined problem into several comprehensive, yet smaller and simpler parts.

The purpose of this article is to propose a specific prototyping method. In order to clarify the relationship between this method and alternative application development methods, we present a state-transition model of the IS development process. Following this, we describe a two-prototype process and our experience with applying it to two IS projects.

A State-Transition Model

Application development and prototyping can be viewed as state-transition processes as shown in Figure 1. In this model each state represents a system, and efforts to change the system are transitions. From a systems life cycle perspective, state #0 might represent the initial system, state #1 through state #N – 1 might represent enhancements and modifications to the initial system, and state #N might represent a mature system. The transitions might be accomplished by various development methods. For example, state #1 might be achieved after a systems development project that resulted in the purchase and installation of a packaged application system. States #2 through #N might result from IS department efforts to expand the packaged system to make it more effective and efficient. Transition #N + 1 could represent efforts to replace an undesirable mature system.

The prototyping approach views the final operational system as the desired state that is achieved by passing through earlier, less desirable states. The transition from one state to the next can be accomplished by the traditional development process of analysis, design, and implementation, or by other appropriate methods. Naumann and Jenkins [16] have described a prototyping model as a four-step process which is repeated until a satisfactory operational system evolves. But this does not clearly show the potential relationship between prototyping

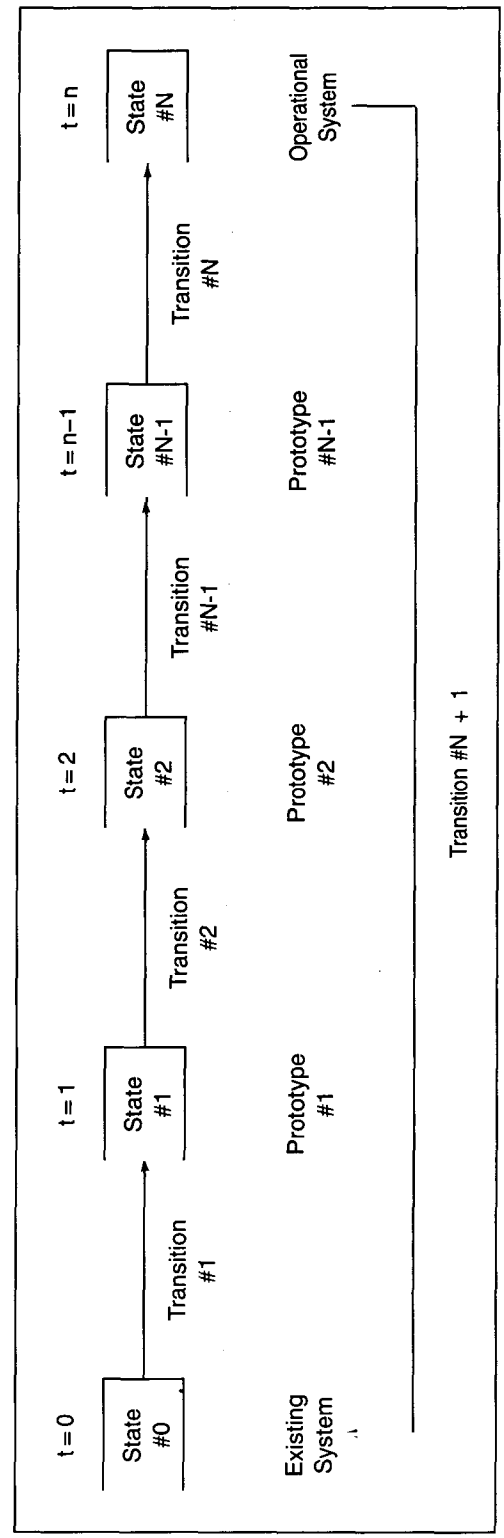

Figure 1. A State-Transition Model for Application Development and Prototyping

and alternative development approaches within the same project. The state-transition model shows that each transition requires a development approach, but not all transitions need to use the same approach.

Another view of the prototyping process might emphasize the states rather than the transitions. An initial working model (state #1) of the target system is first developed. This initial state, or prototype, might represent an incomplete and much simplified model of a dimly perceived operational system. Based on the experience of the users and designers with the initial prototype, a second and more complex prototype (state #2) can be developed. The process continues in an iterative and adaptive fashion until an operational system (state #N) evolves.

The traditional life cycle and heuristic development methods can be viewed as special cases of the general state-transition application process model. As seen in Figure 2, they have only two states, the current system and the desired operational system. In the life cycle approach as described by McKeen [15] and Zelkovitz [20], a linear sequence of steps is followed to make the single transition. A variation of this method described by Bally, Brittain and Wagner [2] uses a loopy linear sequence to make the transition. Berrisford and Wetherbe [5] have suggested a modification to the traditional life cycle process called heuristic development. They propose the use of prototypes in the design step of the traditional development cycle, however it should be clear from Figure 2 that the application of prototypes need not be limited to the design step.

A Two Prototype Development Process

While the literature describes prototyping in general terms, it does not provide operational recommendations for building application systems. In this section we present detailed guidelines for a two-prototype development method by describing our experience with two system projects.

Two different application system development efforts used a two prototype development process. One project developed a large application system for a state child nutrition and development agency. This agency had a budget in excess of $600 million per year with several hundred full-time employees. The project required $250,000 and three years to complete. It used a large IBM mainframe computer and the RAMIS II language.

A second project developed an application system for a small community relations department within a large, high-tech manufacturing facility. This department had an annual budget of less than $100,000 with four full-time employees. It required $31,000 and nine months to complete. An IBM personal computer and a database language for micros called Knowledgeman were used in this project.

The child nutrition and development system project is representative of the types of projects in which IS departments are heavily involved while the community relations project is more representative of end-user developed application systems where the IS department primarily provides advice and guidance. Prototyping proved beneficial in both of these development projects despite their obvious differences.

The strategy used in both projects required the development of two prototypes and three transitions. The initial prototype provided the users/designers with an application framework. This framework was used to explore the users' needs with respect to an operational system. The emphasis at this point was on identifying and experimenting with the output components of the prototype in order to create an operational system.

The second prototype was developed to further explore the output needs of the users, identify and experiment with the system input requirements, and evaluate efficiency considerations. The emphasis was on identifying and experimenting with components of the prototype in order to make the operational system both effective and efficient.

Each of the three transitions required a modification of the traditional linear development methods. The first two transitions included an additional step between testing and evaluating that allowed users and designers to experiment with the prototype. Another modification of the

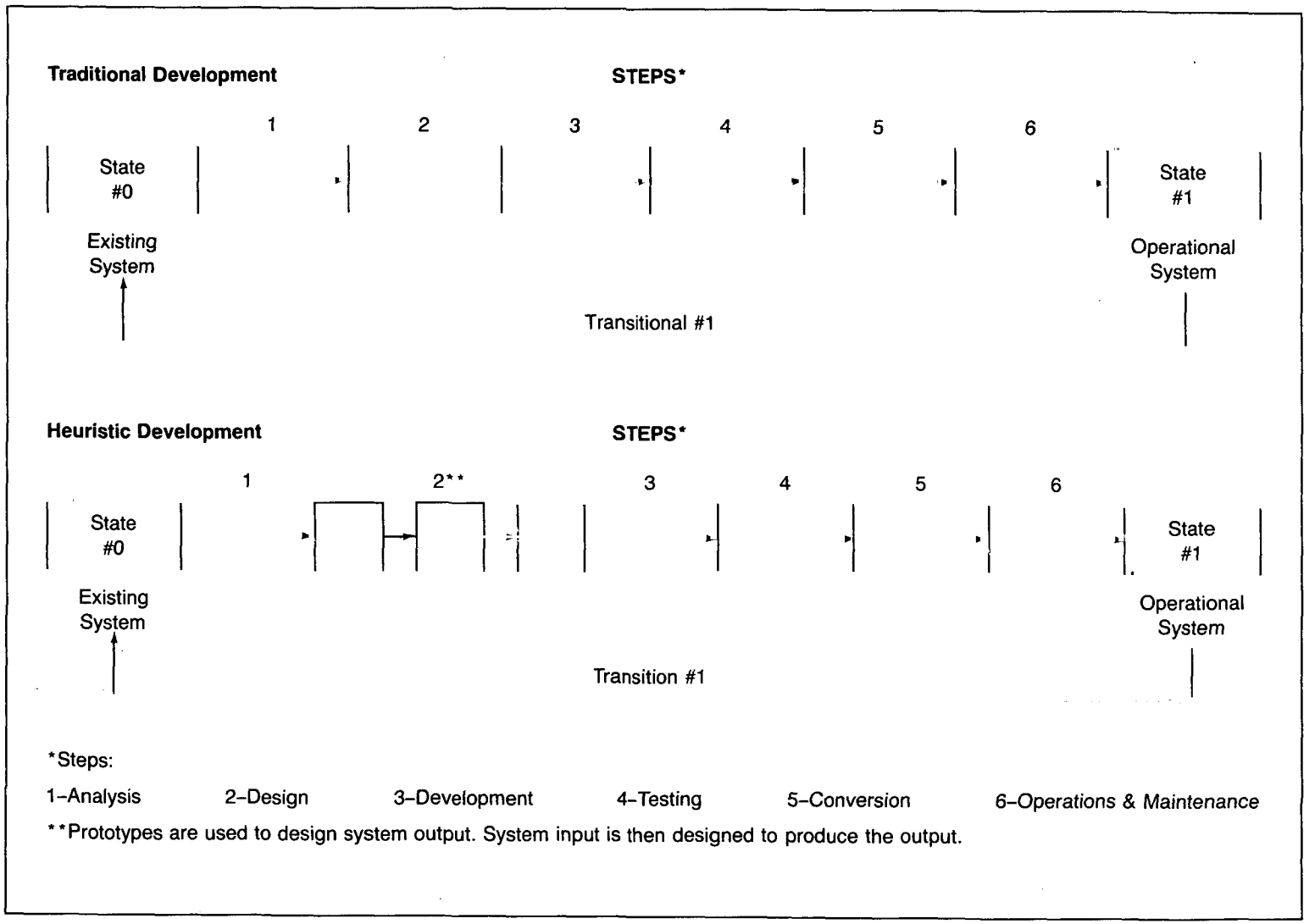

Figure 2. State-Transition View of the Traditional and Heuristic Application Development Processes

traditional development approach occurred in the evaluation step. Since evaluation of the prototype occurred *during* the project, it became an integral part, rather than a mere formality as in the traditional development method in which evaluation occurs at the end. The third transition did not require an analysis step since virtually all of the analysis was performed in the first two transitions.

Our prototyping strategy used essentially the traditional linear development method for each transition. However, the scope of each prototype was much less than the operational system so that each transition proceeded quickly. The strategy was very pragmatic. The goal was to produce useful systems quickly and efficiently.

Transition #1: Building an initial prototype

Needs Assessment

The needs assessment for initial prototype development balanced the time spent identifying and prioritizing user needs with the requirements to build an operational prototype quickly. Recognizing that a complete needs assessment was not possible within the time constraints, a framework of profiles and reports was identified. A profile is basically a snapshot of a single occurrence of an entity at one point in time while a report describes a collection of occurrences for one or more entities, either at one point in time or longitudinally. In a later step, changes to the contents of these initial profiles and reports, along with the development of additional ones, were a means for communicating the prototyping philosophy of adaptation based on experimentation.

Design of the Initial Prototype

Our experience and previous research findings indicate that users relate quickly and easily to systems composed of a series of linked screen menus [5, 14]. These menus provide a logical framework for profiles and reports that comprise the system. A simple illustration of these linked menus is provided in Figure 3.

A sample of data items (fields) was selected for each of the profiles and reports that were chosen for prototype development in step 1. The data items were representative of the type of information users expected and had values that were easily obtained.

After defining the prototype output, the database was designed. There were several acceptable models for the logical design of the prototype database. A semantic database model could provide a tool for expressing the meaning and structure of data. An entity-relationship model could be used to emphasize entities and relationships between entities. A relational model would use tables or flat files to represent relations. Common attributes would relate the tables and allow for much flexibility as relationships need not be predefined. We chose the relational data model for the design of the logical database because its flexibility allowed us to make changes quickly.

The physical design was different for each project as they used two quite different DBMSs. In both projects, attempts were made to normalize the logical relations before the tables were defined in the DBMS. This normalization process is extremely important as it provides the prototypes with the flexibility necessary for rapid change.

A data sample was collected from users and loaded directly into the database via a utility provided by the DBMS. Efforts were made to collect "real" data that represented what the users considered to be typical values. A recent attempt in another prototyping project to use data "created" by the designers was not successful as users spent more time and effort trying to relate to the data than in evaluating the prototype. The sample of data was small with 1-5 occurrences for each data item.

Implementing the Initial Prototype

Using the screen formatting and report writing features of the DBMS, the prototype application program was created. The process of creating the initial program could be accomplished quickly since the language was already known.

Testing the Initial Prototype

The correctness of the application program was determined by adequate testing with the

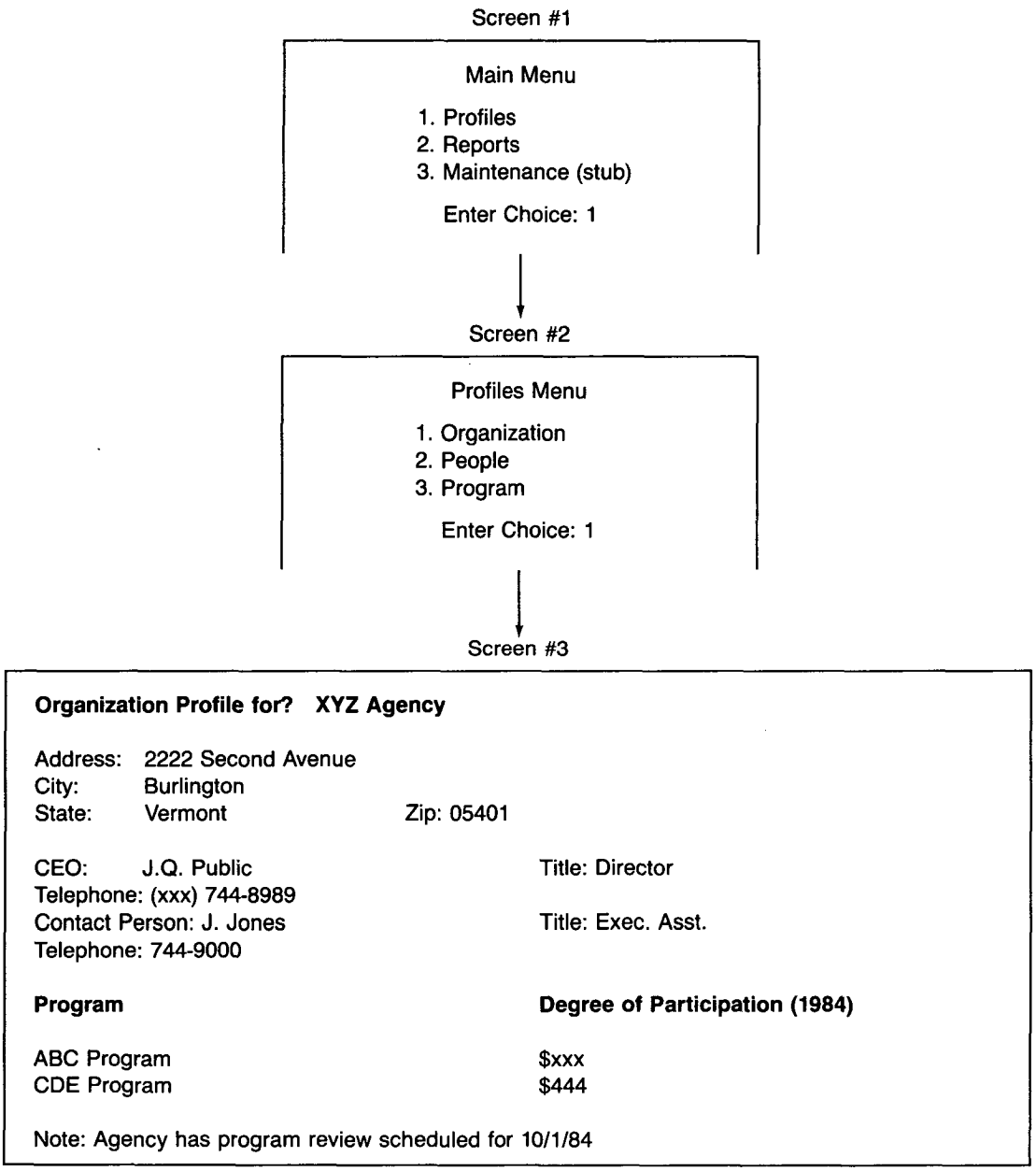

Figure 3. An Illustration of Linked Menus for Initial Prototype

designers and a single representative user. This was a relatively easy task because of the simplicity of the inital prototype and the use of the DBMS screen formatting and report writing features.

Experimenting With the Initial Prototype

After a short presentation on the project and the prototyping process and philosophy, the prototype was demonstrated to small groups of 1 to 4 users. The purposes of this demonstration was to set the "experimental" tones and to give the users an understanding of the prototype's general framework.

After the short demonstration, users were encouraged to suggest changes and additions to the system with respect to menu organization, menu contents, and profile and report contents. Those suggestions that seemed representative and simple were made immediately utilizing the power of the DBMS system. These "on-the-fly" changes further encouraged the experimental tone and user involvement. Representative changes requiring more substantial effort were proposed for inclusion in the next prototype. This step was designed to utilize the prototype as a vehicle to bring users into the development effort, revise the users previous needs assessment, and communicate project purpose and goals.

Evaluating Initial Prototype and Project

Based on the experience obtained from building and exercising the initial prototype, a proposal for a second prototype was developed. Management evaluated this proposal as if it were an entirely new project. The cost of developing the next prototype was weighed against the potential benefits of an operational system. Keen [12] presents a method for determining if the cost of a project is justified when benefits are qualitative.

Transition #2: Building the second prototype

A design process similar to the design of the initial prototype was followed. However, the design extended and/or modified the initial prototype to develop a more realistic model of the desired operational system with respect to both effectiveness and efficiency factors. The second prototype included the following:

- illustrative database loading and maintenance routines for users,
- novice (linked menus) and expert (function keys) usage modes,
- software monitors for recording prototype usage,
- more and expanded reports and profiles, and
- a more efficient database.

The same sample of user groups that experimented with the first prototype continued to experiment with the second. However, whereas the designers operated the initial prototype, the users operated the second prototype after a short demonstration of novice and expert usage modes.

The initial sample of users was expanded to include users with responsibility for the loading and updating of the data, and representatives of the IS department management, if they were not already part of the project team. This insured that groups having an interest in the project were exposed to the prototype during at least one of the first two transitions. For example, with the community relations project representatives (management and development staff) of the IS department were shown a demonstration of the second prototype so that they would be in a position to provide input at the project evaluation step. In the child nutrition and development project, managers and staff representing organizational levels above those directly involved in the project, were shown the second prototype. The purpose was to demonstrate: (1) what had been accomplished quickly with limited resources, (2) the potential of the application system if and when it was operational, and (3) the prototyping process and its advantages and disadvantages.

The usage of the prototype was recorded by the users and the software monitors. Prototype system problems and user/designer suggestions were documented. However, unlike the initial prototype, suggested changes were not implemented unless they were critical to continued successful experimentation of the prototype. Designers also experimented with the prototype to collect operating data. Utilizing this data,

design criteria values for an operational prototype were developed for inclusion in the proposal for an operational system. For example, the anticipated number of concurrent users, records, and record accesses were estimated from the prototype experiments. The adequacy of the system users manual was also determined.

The project team prepared a report that included user/designer experience with the prototype and a proposal for an operational system. The proposal included an estimate of the amount of modification of the prototype necessary and/or desirable for an operational system, a revised list of cost/benefits, and a budget and schedule for transition #3. A presentation to management included the proposal and a demonstration of the prototype.

Transition #3: Building the operational system

The effort required for this transition is proportional to the degree of modification required of the second prototype. In some cases the second prototype is very close to an operational system. In others, a relatively full-scale application development project may be required. In either case, the prototyping experience is used to build the operational system.

Many components of the second prototype were transferred to the operational system making the development effort much shorter. For example, some data, software, and data structures were transferred. At the very least, users and designers had a much better understanding of what the operational system should do and how it would operate. Under these conditions the traditional approach was the most appropriate for quickly building the operational system

Summary and Discussion

In this article we have presented a general state-transition model of the application development process that clearly shows the large number of potential application development strategies. We have shown that application development methods such as prototyping and heuristic development can be viewed as transitions between system states. Also we have shown how the traditional life cycle development method can be represented by this model.

The prototype development process appears to be a useful strategy for efficiently delivering effective application systems. In our study, the initial prototype encouraged the development of an effective system by emphasizing the building of a user needs framework. Users were involved in the design phase and were encouraged to experiment with the system in order to define their needs. This led to reasonable system expectations, fewer surprises, and user ownership of the final operational system. The process also accommodated the changes typically experienced in the user's dynamic world and allowed the projects to focus on building and expanding rather than limiting and controlling. The second prototype was used to further explore and develop the system definition while processing efficiency considerations were examined. It gave the users and designers a realistic view of the potential effort required to operationalize and maintain the system and helped insure an efficient operational system.

It would appear that the prototype process would not be appropriate when user needs are static or well-defined, or when development experience with similar applications has been extensive. Of course, more research and experimentation is needed to determine which applications are not well suited to prototyping. However, our experience indicates that prototyping can provide on-time and within-budget systems for both large and small application projects that are typically developed by information systems specialists and/or end users. The prototyping methodology clearly has the potential to improve productivity over a wide range of applications.

References

[1] Bally, L., Brittan, J., and Wagner, K.H. "A Prototype Approach to Information System Design and Development," *Information and Management,* Volume 1, Number 1, November 1977, pp. 21-26.

[2] Behrens, C. "Measuring the Productivity of Computer Systems Development Activities

with Function Points," *IEEE Transactions on Software Engineering,* SE-9, Number 6, November 1983, pp. 648-652.

[3] Benson, D.H. "A Field Study of End User Computing: Findings and Issues," *MIS Quarterly,* Volume 7, Number 4, December 1983, pp. 35-45.

[4] Berrisford, T., and Wetherbe, J. "Heuristic Development: A Redesign of Systems Design," *MIS Quarterly,* Volume 3, Number 1, March 1979, pp. 11-19.

[5] Brown, J.W. "Controlling the Complexity of Menu Networks," *Communications of the ACM,* Volume 25, Number 7, July 1982, pp. 412-418.

[6] Canning, R.G. "Developing Systems by Prototyping," *EDP Analyzer,* Volume 19, Number 9, September 1981, pp. 1-12.

[7] Canning, R.G. "Where Will Applications Be Developed," *EDP Analyzer,* Volume 21, Number 12, December 1983, pp. 1-12.

[8] Gill, H., Lindvall, R., Rosin, O., Sandewall, E., Sorensen, H. and Wigertz, O. "Experience from Computer Supported Prototyping for Information Flow in Hospitals," *ACM SIGSOFT Software Engineering Notes,* Volume 7, Number 5, December 1982, pp. 67-70.

[9] Howden, W.E. "Contemporary Software Development Environments," *Communications of the ACM,* Volume 25, Number 5, May 1982, pp. 318-329.

[10] Jenkins, A.M. "Prototyping: A Methodology for the Design and Development of Application Systems," *Discussion Paper #227,* School of Business, Indiana University, Bloomington, Indiana, April 1983.

[11] Katz, D. and Kahn, R. *The Social Psychology of Organizations,* (2nd Ed.), John Wiley and Sons, New York, New York, 1978.

[12] Keen, P.G.W. "Value Analysis: Justifying Decision Support Systems," *MIS Quarterly,* Volume 5, Number 1, March 1981, pp. 1-16.

[13] Martin, J. *Applications Development Without Programmers,* Prenctice-Hall, Englewood Cliffs, New Jersey, 1982.

[14] Mason, R.E.A., and Carey, T.T. "Prototyping Interactive Information Systems," *Communications of the ACM,* Volume 26, Number 5, May 1983, pp. 347-354.

[15] McKeen, J. D. "Successful Development Strategies for Business Applications Systems," *MIS Quarterly,* Volume 7, Number 3, September 1983, pp. 47-65.

[16] Naumann, J.D., and Jenkins, A.M. "Prototyping: The New Paradigm for Systems Development," *MIS Quarterly,* Volume 6, Number 3, September, 1982, pp. 29-43.

[17] Rockart, J.F. and Flannery, L.S. "The Management of End User Computing," *Communications of the ACM,* Volume 26, Number 10, October 1983, pp. 776-784.

[18] Scott, J. "The Management Science Opportunity: A Systems Development Management Viewpoint," *MIS Quarterly,* Volume 2, Number 4, December 1978, pp. 59-61.

[19] Young, T.R. "Superior Prototypes," *Datamation,* Volume 30, Number 7, May 1984, pp. 152-158.

[20] Zelkowitz, M.V. "Perspectives on Software Engineering," *ACM Computing Surveys,* Volume 10, Number 2, June 1978, pp. 197-216.

About the Authors

James M. Kraushaar *is an Associate Professor in the School of Business Administration at the University of Vermont. He received an M.S. in Industrial Engineering Administration and a Ph.D. in Business Administration from Syracuse University. He has published in professional journals and proceedings and is the author of a textbook on structured BASIC. Dr. Kraushaar has served as a consultant to the California Department of Education, IBM, U.S. Departments of Agriculture and Interior, and numerous small corporations and nonprofit organizations. His research interests include application development methodologies and computer modeling.*

Larry E. Shirland *is an Associate Professor in the School of Business Administration at the University of Vermont. He received his M.S. and Ph.D. degrees in Industrial Engineering from Oregon State University. He has published in professional journals including the* Academy of Management Journal, Decision Sciences, Interfaces, *and* Omega. *Dr. Shirland has consulted for firms which include Anheuser-Busch and IBM, and was a project engineer with Eastman Kodak. His research interests include operations scheduling and mathematical programming applications.*