

# END-USER PROTOTYPING: SOPHISTICATED USERS SUPPORTING SYSTEM DEVELOPMENT

By Nava Pliskin and Peretz Shoval

*An accepted assumption underlying application development is that the user is naive, with little understanding of data processing. All the methods for handling the information system development process, from structured analysis to prototyping, have in common a concern with user/developer communication, yet always under the assumption that users are alike in their capacity to understand data processing.*

*This paper proposes to examine this preconception of the user. With the prevalence of end-user computing, users should no longer be considered universally ignorant of information systems. Moreover, sophisticated information center veterans, as well as microcomputer users, can and do contribute to system development. They are capable of preliminary experimentation with system requirements, in particular those related to user interface. Thus, the concept of end-user prototyping is developed and demonstrated, presenting information systems professionals with an opportunity to profit from enhanced user sophistication.*

---

## INTRODUCTION

---

System development, even today, is still not pursued according to a universal prescription. The available techniques are usually either of the structured life cycle type or of the prototyping type; sometimes they are a hybrid of both types [3]. Among the various existing approaches to information systems (IS) development, prototyping is a controversial one [8] with conflicting reports on its effectiveness and efficiency. Prototyping seems to be appropriate for small problems and is especially useful in determining interaction requirements [18]. For complex problems, however, prototyping increases the project risk when compared with structured life cycle techniques [12].

We question the assumption, common to all system development methods, of equality among users in terms of data processing understanding, since microcomputers and information centers have caused the upgrading of some users from naive to sophisticated. This paper brings together rapid prototyping and end user computing to define end-user prototyping. IS developers are advised to take advantage of the increased sophistication of users and are provided with demonstrations in support of the idea.

---

## SYSTEM DEVELOPMENT

---

Structured development life cycle (SDLC) approaches to IS development have been in practice for more than two

---

*Drs. Pliskin and Shoval can be reached at the Ben-Gurion University of the Negev, P.O. Box 653, Beer-Sheva 84105, ISRAEL.*

decades. Though the literature provides many definitions and methods, SDLC is quite standard [1]. SDLC is highly structured, consisting of the following phases: preliminary survey, analysis, design and creation. Systems then move to stages of implementation and operation which involve mainly maintenance and enhancement.

SDLC techniques are applied by a professional working alongside users who are expected to provide and specify their information requirements at the beginning, as well as to approve the system upon completion. Available SDLC tools for analyst-user communication are mostly graphical or textual representations that are neither effective nor efficient. Hence, even upon approval of the design, both user and analyst may inadequately understand the system requirements. Moreover, since SDLC involves a requirement "freeze" between design and implementation, environmental dynamics may be taken into account, resulting in an unsatisfactory or even useless system from the user's point of view [7]. Thus, according to Wasserman [24], user dissatisfaction is a major issue of concern with SDLC.

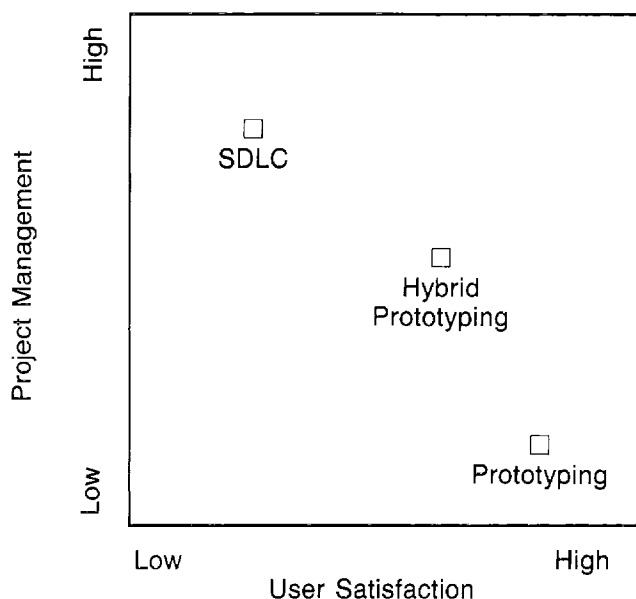
SDLC drawbacks, in particular user-related ones, triggered the concept of prototyping borrowed from other engineering disciplines [26]. No universal definition of prototyping exists and different terms are used for referring to it [2, 5, 6, 14, 21]. The following definition, based on Sprague and McNurlin [25], is adopted here: An iterative process of creating quickly and inexpensively live and working models to test out requirements and assumptions. According to Keus [24], if the process results in an actual production system it is called "add-on" or evolutionary prototyping. If, however, the prototype is discarded and a production system is constructed, it is called throw-away prototyping.

In prototyping, as in SDLC, a builder (acting as an active professional) and a user (in a passive and naive capacity) cooperate to reach a working application [21]. The means of communication are demonstration models that the builder develops quickly using a fourth generation language. Such models are real enough, and present the user with an approximation of the system rather than an imaginary one [13], thus enhancing communications.

Although many success stories on prototyping have appeared in IS literature [11, 14, 15, 17, 27], there has been criticism as well [8, 12]. Prototyping is associated with more user satisfaction and less cost than SDLC, but it is also known to result in less efficient systems and development processes that are more difficult to manage and control [2, 3]. This critique is most relevant to evolutionary prototyping which can lead to poorly documented systems. It is not surprising, therefore, that IS professionals also mix SDLC and prototyping. In recent years the IS literature has focused on the utilization of a hybrid process [7] where prototyping is used for requirements analysis while the remainder of the development process uses SDLC. Throw-away prototyping can be a part of a hybrid prototyping approach. Experience with

this approach shows that it compensates for the drawbacks in its parent methodologies.

A decision to adopt SDLC, prototyping, or a hybrid approach has multiple implications from an IS management perspective. Value tradeoffs are involved [21]. Prototyping is rated higher for both user satisfaction and cost, while SDLC is higher on final product efficiency as well as management and control of the development itself. For a graphical illustration, consider only two dimensions, user satisfaction and project management and control. In Figure 1 below, placement of the three development alternatives is in line with the discussion thus far. Though not expected to exceed its parent methodologies on their advantageous dimensions, hybrid prototyping should perform better than prototyping on project management, and better than SDLC on user satisfaction.



**FIGURE 1. System Development Value Tradeoffs**

The IS organization is no longer expected to employ a single IS development method. It is called upon rather to design a development strategy aimed at matching the development approach with specific application characteristics. Generally, enhancing the degree of design formality is recommendable (that is, more emphasis on SDLC) as systems become larger and more complex [10]. Prototyping should be introduced as the degree of user interactions and project uncertainty (in terms of user requirements) increases [7, 17]. IS management is expected to apply value judgment in choosing the development alternative and to be aware of the multiattributed nature of the decision.

**DEVELOPMENT STRATEGY AND THE USER**

The IS literature recognizes the need for data processing (DP) management to explicitly define a system development strategy and to this end contingency approaches for selecting application development methodologies are available. The decision-making parameters are application-related

rather than user-related. In other words, the type of user is not considered as an attribute or an integral part of the decision problem. There is an implicit assumption that all users are equal in terms of DP literacy. All system development methodologies assume a situation in which professional developers confront naive users devoid of DP skills. However, in the highly technical environments of today one can no longer assume users to be totally naive. The experience of end-user computing raises a person from the category of computer illiterate [23]. The same is true for graduates of educational environments that familiarize students with computers.

In recent years the IS community has recognized that users can no longer be thought of as universally naive. A few user categorization schemes have been proposed. For example, Rockart and Flannery [23] propose a six-category view of users: non-programmer end users, command level users, end-user programmers, functional support personnel, end-user computing (EUC) support personnel and DP programmers.

For our purposes, Martin's [16] classification is sufficient. He observes three user types: non-programming end users, DP amateurs and DP professionals. We offer, in Table 1, to view Martin's categorization in a two-dimensional framework. The dimensions are programming skills (can the user program?) and DP professionalism (does the user exhibit expertise and experience in DP?). Historically, DP professionals developed from programmers and therefore the bottom left corner of Table 1 is empty. This leaves in the left column of Table 1 one relevant box which, in the context of system development, is populated by IS builders. In data processing all programming activities had been in professional hands until information technology brought about end-user computing and made feasible the combination of proven programming skills without DP background. These users, whom we call sophisticated, are Martin's DP amateurs. Finally, we view non-programmer end users as naive with respect to system development.

Beyond the two-way classification of Table 1, it is possible to elaborate further on the sophisticated and naive user types. In the former group are end users who apply information technology independently from the IS organization. They go beyond keying data into applications provided for them by the IS organization. Some access the central data resource using the query facilities of a database management system. Others develop applications using fourth generation languages. Users are considered naive if they have no access to a computer, or if their exposure to information technology through applications is so transparent that there is no value added in terms of data processing understanding.

Sophisticated users are a product of the last decade. They have either some formal computer training (for example, as graduates of engineering and business schools) or they have

**TABLE 1. Types of End Users**

		DP PROFESSIONALISM	
		Yes	No
PROGRAMMING SKILLS	Yes	IS Builders	Sophisticated
	No		Naive
		PROFESSIONALS	USERS

been autodidactic in overcoming computer illiteracy. They have no fear of technology and can utilize it to support their professional work when management provides the appropriate tools.

The benefits of this trend, according to [16], are in reduction of application development backlogs and IS maintenance loads, but risks emerge [20]. Having no real DP credentials, end user programmers produce user developed applications (UDAs) that are deficient in documentation, backup, security and integrity [22]. Though IS management can help by providing professional assistance and guidance [14], UDA management is problematic and should be handled carefully [26]. Similar risks also afflict less sophisticated forms of EUC, such as application of microcomputer software for professional support.

Conscious of the above risks, IS management is usually concerned with the management of EUC in general and UDA in particular [23]. This concern of the IS organization is related to its control of the information environment and is usually separated from the issue of system development. However, we recommend avoiding such separation and suggest instead the channeling of accumulated EUC experience to support the system development process. In other words, the system development strategy question is broadened to account for the user type, as well as the application type, in selection of the IS development methodology.

### End-User Prototyping

Two roles are usually mentioned in relation to prototyping: user and builder [13]. Mason and Carey [17] developed the role of an architect—an IS professional prototyping the user's requirements to ease communication with the builder. Introduction of an architect role suggests perhaps that a user is necessarily unsophisticated.

If sophisticated users become involved in the system development process in a conventional naive capacity, opportunities for process improvement are likely to be missed. The alternative is to incorporate end-user development into IS development [6, 15, 19]. The definition of end-user prototyping (EUP) makes explicit the changing role of the user and is applicable only to a sophisticated user. EUP is the iterative, quick and inexpensive creation (by a sophisticated user) of live, working models to define and test system requirements. EUP is different from both standard hybrid prototyping, since under either of the latter development strategies users are assumed to have no DP knowledge and to take less participative user roles [21].

Because of user sophistication, EUP might be initiated before the IS organization is even aware of the need for a system. It would be unrealistic, therefore, to expect or to dictate involvement of the IS organization from the beginning of a project. Still, the IS department would be well advised to monitor the user population for EUP startups as well as to assign an IS professional to oversee the process. This liaison person should advise IS management on the maximum advantage to be drawn from user contributions to the further development of the actual production system.

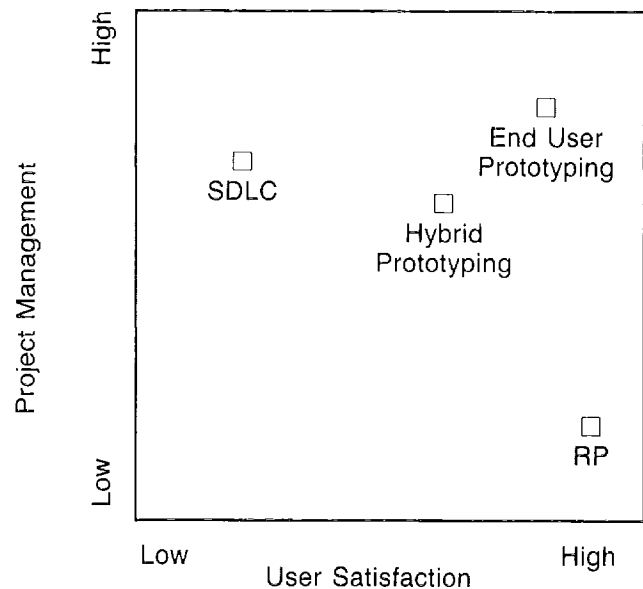
Within this conceptual EUP framework two sub-categories exist, depending on whether development is completed by the user or not. If evaluation of the prototype [9] by IS management indicates that the user is capable of carrying the development process toward completion, an IS profes-

sional could be assigned to provide professional support, resulting in an IS-controlled UDA and reducing the risks mentioned previously.

If IS management considers the development task to be beyond the user's ability, an IS builder could assume development responsibility while still allowing prototyping contributions. This development mode is a modified hybrid approach where a sophisticated user performs part or all of the prototyping.

In summary, EUP is a system development methodology that takes advantage of user sophistication and ends up in either a professionally supervised UDA or in a production system developed by the IS organization. End-user programming is thereby a strategy that distinguishes itself from other development methodologies. In the case histories detailed in the next section end-user programming performed better on relevant decisions attributes. At this stage we can only hypothesize the advantages based on our experience and acknowledge that the dominance of end-user programming has yet to be conclusively ascertained. The following paragraphs include several logical arguments in support of the dominance hypotheses.

When IS professionals engage in prototyping project management becomes difficult, but in prototyping by end users it might be possible for the builder to manage the development project as effectively as with SDLC. Also, because in end-user programming the user is more active and involved, the score on user satisfaction is probably better than with system designer prototyping. In Figure 2, end-user programming is placed accordingly in dominance over the development options.



**FIGURE 2. System Development Value Tradeoffs with End User Prototyping**

Opponents of prototyping view it as a "cover-up" for a sloppy development that may result in a narrow design focus, operational difficulties [7], a general reduction in the development discipline, and overall poor planning, budgeting and control [8]. In end-user programming the IS builder can avoid sloppiness because it is mainly the user who pursues prototyping. Furthermore, the builder can assume a su-

pervisory role and emphasize discipline, planning, budgeting and control.

The impact of EUP on the investment of user time in the project has yet to be evaluated and, if warranted, justified. In the case studies examined later, time spent on prototyping by sophisticated users meant savings along the life cycle of the system through better communication with the IS organization and increased user satisfaction. In addition, time spent by users lead to a decrease in time invested by IS professionals and to improvements in system development bottlenecks. Thus, the organization might view the benefits from EUP to be of enough value (compared to other system development strategies) to justify user time expenditure.

### *Demonstration Case Studies*

**T**he case studies that follow provide evidence of real benefits received in an international chemical production organization using this concept. End-user computing was commissioned in 1984, when the proliferation of microcomputers was already phenomenal, by a committed and supportive top management. The company was growing at a rapid pace and the need for personal computing had become apparent. In particular, the vice president for marketing felt that in the absence of centrally developed marketing applications personal computing was essential to a successful performance internationally, especially at a time of market expansion and increased competition. IS management was also supportive in identifying the opportunity for selective distributed data processing to relieve some of the pressure exerted by marketing on the resources of the IS group.

Recommendations on end-user computing management in leading publications [23] have been implemented. An outside training group was hired for the purpose of introducing users to the technology. A functional support position was created so that users could get in-house help and advice. Users were encouraged to share personal computers and that created a positive, non-threatening environment. Within six months, twenty-five users had access to seven microcomputers, printers, a plotter, Lotus 1-2-3, dBASEIII, word processing and software to handle such applications as statistics, library management and communication.

Eighteen months of end-user computing found users, superiors, and IS personnel to be satisfied with the quality of work life in the era of personal computing. IS management noted a relief in marketing pressure to deliver centrally developed applications, more understanding of MIS difficulties on the part of users, and improved communication between the IS and marketing groups.

These positive attitudes both before and after end-user computing introduction provide the background for two EUP case studies. In both cases, prototyping activities were limited to a sophisticated minority. These users came forward during interviews to determine requirements for end-user computing and expressed avid interest in the technology. Most had IS backgrounds either by education or as users of company IS. Soon after they had the tools at their disposal, and even before completion of formal training, they had developed applications to replace manual tasks and proceeded to computerize tasks that were not feasible manually. They were friendly, voluntarily sharing applications with and providing support to peers. Other characteristics observed were: professionalism in their field, analytic skills, improvisation capacity, political and organizational awareness.

In the first case of EUP, prototyping was not intended. A sophisticated user, believing that the microcomputer could do everything, interviewed other relevant users and attempted to develop a client database using dBASEIII. Soon afterwards memory limits and the need to share the database internationally brought the user to the IS organization requesting central development.

The activities up to that point, and user interaction with the IS professionals afterwards, meet the definition of EUP. The preliminary communication with the IS professionals was based on the requirements of a user group, presented by means of the dBASEIII prototype and yielding a very effective requirement definition phase. Both users and IS professionals considered the remainder of the development process a relatively positive experience that included conventional SDLC activities in parallel to continued EUP.

The second EUP case had a completely different evolution. Users had developed a spreadsheet application for computerization of the monthly production planning process. Tedious manual calculations were replaced by a Lotus 1-2-3 application that involved two stages. First, suppliers of relevant data presented it on diskettes to a planning coordinator who then combined the data and brought a preliminary production plan into a planning meeting. There, the final plan was completed after on-line sensitivity analyses.

This production planning support system was operational very soon after the introduction of personal computing and was constantly improved upon and modified by users. For a long time it was considered by both users and IS professionals as an EUP that matured into a UDA. After two years of operation, despite continued successful performance, users expressed preference for a decision support system on the mainframe. In consultation with IS professionals, and with the UDA serving as a preliminary prototype, it was concluded that a central system with access to historical data, interface to neighboring systems, and communication to remote locations was needed. In the development of the requested system, IS professionals used structured development tools whilst the frame of reference in communication between builders and users was based on the line and working models prototyped by users.

The two case histories are quite different. IS professionals expected the evolution of the first system into a classic EUP example, whereas the second constituted a surprise, having been expected to end up as a UDA, but brought back to EUP mode for mainframe development. Still, both cases evolved into throwaway prototypes with identical EUP implications.

Users and IS professionals viewed the EUP experience as being distinct from any other design strategy they had experienced. There was also evidence of real benefits to the company from the final products being centrally designed as they become strategically better and implementation was smoother. An additional organizational advantage was in the changing roles and relationships within the company. The sophisticated user became a liaison officer between a group of naive users and IS personnel, thus rendering communication within the organization more meaningful. Finally, expenditure of user time in both cases was not excessive relative to other development methods.

Prior to EUP, users had to spend time in thinking about requirements both before and during communication with IS professionals. With EUP the preliminary thinking took

place while developing the prototype. Actual programming framed the productive thinking process, similar to Lotus 1-2-3 improving the paper-and-pencil alternative. The investment of work up to that point was compensated for by savings in communicating the requirements. For example, in the client database case, after experimenting with dBASEIII users knew very well the information content of fields and their lengths, the inputs and outputs, and the functions requested, and offered solutions to cope with the complexities of uniqueness in international coding.

Another element of savings was in user group time. The IS professional communicated throughout the life cycle with the prototyping users who represented the group. This was more efficient than communication between the developer and the entire group.

The development was also faster than usual because many obstacles had already been met and solved by users in the prototyping phase, leaving the professional with a more logical and consistent set of requirements. The resulting shortcuts in the development sequence saved both user and professional time. If EUP actually claims an increased expenditure of end user time, the savings in IS professional time would still be more than compensatory since this resource is scarce and limited.

### Appropriate Uses for End-User Prototyping

After presenting four alternatives to system development, including EUP, the discussion now relates the system development approach to a typology of information systems. For the discussion here, the classification consists of three information systems types: a) small, b) large, and c) packages. The first two types are tailor-made for a specific customer, whereas the third type refers to a mass-produced, standard system. The differentiation between the first two systems is as follows:

- a. Small IS: Such systems can usually be developed on a local computer, serving one or several users, and are "light" in terms of storage volume, transactions load, and logic. The usage of application generators is feasible.
- b. Large IS: Such systems are usually implemented on mainframes, serving many users, and are "heavy" in terms of storage volume, data structure complexity, transaction load, and logic. The use of application generators and fourth generation languages may not be feasible because of performance considerations.

From a system development perspective, small systems can be developed with no precise distinction between phases of development, whereas effective development of large systems requires such distinction. Thus, while structured techniques are not essential to the development of small systems, they are mandatory for large ones.

For small ISs, a sophisticated user can handle many development activities by means of prototyping, with possible evolution of the prototype to the system itself. Prototyping is applicable under both user types; performed by a builder if the user is naive (RP), and in the form of EUP if the user is sophisticated.

For large systems, user prototyping will have to be relatively limited. For the most part, prototyping can only be utilized in the requirement definition phase for determining inputs, outputs and other user-system interfaces. Eventually, and irrespective of whether user, builder, or both are prototyping, it is a throw-away situation and the application of structured methods is required for project completion. Thus, for large systems development, if naive users are involved a hybrid approach can be carried out by a builder. In the environment of sophisticated users, however, a modified prototyping approach can take place in the form of EUP.

Unlike the first two types of ISs, packages are developed by vendors for mass marketing. There is virtually no real user to work alongside the developer so the SDLC approach is appropriate. Before marketing the product, experiments with a trial version are usually carried out at beta sites. Thus, the SDLC produces a trial system which is tested by customers who, through experience, contribute to the enhancement of the product.

The different types of IS not only dictate different development strategies, but are also associated with different degrees of user involvement. The contribution and participation of users (whether sophisticated or not) diminishes as one moves from small to large systems and then to packages. Though these trends apply to both naive and sophisticated users, the magnitude of difference is higher for the sophisticated user.

The rectangles in Figure 3 represent the contributions of users and builders to system development. The white areas are a reflection of the naive user share, which diminishes along two dimensions. First, the white areas shrink with system size and commercialization. Second, the same trend occurs as the development process moves away from preliminary system development stages toward advanced ones. The dotted areas are the incremental contributions due

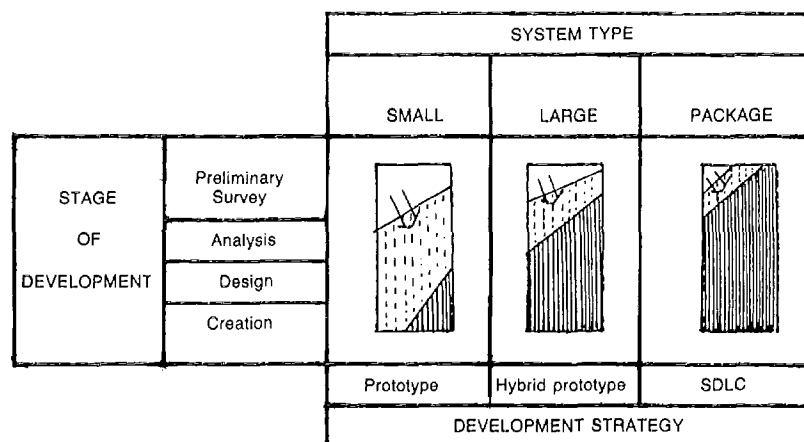


FIGURE 3. The Contribution of Users to System Development

to user sophistication. These increments are substantial for small systems, but are almost negligible for packages.

---

## CONCLUSION

---

End-user prototyping already exists in many IS environments. This paper shows where a development strategy alongside sophisticated users might differ from either traditional or prototyping processes. Further research is needed to evaluate EUP and to ascertain conclusively that EUP benefits exceed costs. This research should be geared to the identification of systems and user environments suitable for this approach. It will be necessary to investigate the interaction and optimal division of contribution between builders and both user types. Research must not be limited to the "how" of the system development dilemma (if, when, and to what extent prototyping is applicable). Rather, the question of "who" must also be addressed: "Who shall carry out the prototyping task: a sophisticated user? a developer? or both?"

So far, practical guidelines to IS organizations on the management of end-user prototyping are not available and the advantages of a sophisticated user are not pursued. Sophisticated users are a precious resource in system development. EUP research will pave the way toward improved management of this resource.

---

## REFERENCES

---

- [1] Ahituv, N., Hadass, M., and Neumann, S., "A Flexible Approach to Information System Development," *MIS Quarterly*, Volume 8, No. 2, June 1984, pp. 69-78.
- [2] Alavi, M., "An Assessment of the Prototyping Approach to Information Systems Development," *Communications of the ACM*, Volume 27, No. 6, June 1985, pp. 556-563.
- [3] Alavi, M., "The Evolution of Information Systems Development Approach: Some Field Observations," *DATABASE*, Volume 18, No. 3, Spring 1984, pp. 19-24.
- [4] Benson, D.H., "A Field Study of End User Computing: Findings and Issues," *MIS Quarterly*, Volume 7, No. 4, December 1983, pp. 35-45.
- [5] Berrisford, T.R. and Wetherbe, J.C. "Heuristic Development: A Redesign of Systems Design," *MIS Quarterly*, Volume 3, No. 1, March 1979, pp. 11-19.
- [6] Bohr, B.H. *Application Prototyping: A Requirements Definition Strategy for the 80s*, John Wiley & Sons, New York, NY, 1984.
- [7] Burns R.N., and Dennis A.R. "Selecting the Appropriate Application Development Methodology," *DATABASE*, Volume 17, No. 1, Fall 1985, 19-23.
- [8] Carey, T.T., and Mason, R.E.A., "Information System Prototyping: Techniques, Tools and Methodologies," *INFOR*, Volume 21, No. 3, August 1983, pp. 177-187.
- [9] Church, V.E., Card, D.N., Agresti, W.W., and Jordan, Q.L., "An Approach for Assessing Software Prototypes," *ACM SIGSOFT Software Engineering Notes*, Volume 11, No. 3, July 1986, pp. 65-76.
- [10] Davis, G.B., and Olson M.H., *Management Information Systems Conceptual Foundations, Structure and Development*, McGraw-Hill Book Company, New York, 1985.
- [11] Earl, M.J., "Prototype Systems for Accounting, Information and Control," *DATABASE*, Volume 13, No. 2-3, Winter-Spring 1982, pp. 39-46.
- [12] Janson, M.A., and Smith L.D., "Prototyping for Systems Development: A Critical Approach," *MIS Quarterly*, Volume 9, No. 4, December 1985, pp. 305-316.
- [13] Jenkins, A.M., "Prototyping: A Method for the Design and Development of Applications Systems," *Spectrum*, Volume 2, No. 2, April 1985.
- [14] Keen, P.G.W., "Adaptive Design for DSS," *DATABASE*, Volume 12, No. 1, Fall 1980, pp. 15-25.
- [15] Kraushaar I.M., and Shirland L.E., "A Prototyping Method for Applications Development by End Users and Information Systems Specialists," *MIS Quarterly*, Volume 9, No. 3, September 1985, pp. 189-196.
- [16] Martin J. *Application Development Without Programmers*, Prentice Hall, Engelwood Cliffs, NJ, 1982.
- [17] Mason R.E.A. and Carey, T.T. "Prototyping Interactive Information Systems," *Communications of the ACM*, Volume 26, No. 5, May 1983, pp. 347-354.
- [18] McFarlan, F.W., "Portfolio Approach to Information Systems," *Harvard Business Review*, Volume 59, No. 5, September-October 1981, pp. 142-150.
- [19] McNurlin, B.C., "Where Will Applications be Developed," *EDP Analyzer*, Volume 21, No. 12, December 1983, pp. 1-12.
- [20] Michielsen K., "Micro Applications Development," *Datamation*, Volume 32, No. 8, April 1986, pp. 96-98.
- [21] Naumann J.D., and Jenkins A.M., "Prototyping: The New Paradigm for Systems Development," *MIS Quarterly*, Volume 6, No. 3, September 1982, pp. 29-44.
- [22] Rivard F., and Huff, S.L., "User Developed Applications: Evaluation of Success from the DP Department Perspective," *MIS Quarterly*, Volume 8, No. 1, March 1984, pp. 39-49.
- [23] Rockart, J.F., and Flannery, L.S., "The Management of End User Computing," *Communications of the ACM*, Volume 26, No. 10, October 1983, pp. 776-784.
- [24] Sprague, R.H. and McNurlin, B.C. *Information Systems Management in Practice*, Prentice Hall, Englewood Cliffs, NJ, 1986.
- [25] Sumner, M.R. "How Should Applications be Developed? An Analysis of Traditional, User, and Microcomputer Development Approaches," *DATABASE*, Volume 17, No. 1, Fall 1985, pp. 25-34.
- [26] Taylor, T. and Standish, T.A. "Initial Thoughts on Rapid Prototyping Techniques," *ACM SIGSOFT Software Engineering Notes*, Volume 7, Number 5, December 1982, pp. 160-166.
- [27] Young T.R., "Superior Prototype," *Datamation*, Volume 30, No. 7, May 1984, pp. 152-158.

# Appendix

## End User Questionnaire

Name \_\_\_\_\_

Job Title \_\_\_\_\_

Work Phone \_\_\_\_\_

**A. Company Environment**

Using the 1 to 5 scale provided, please assess the place of computing in your organization's business strategy

1	2	3	4	5
Information systems support operational efficiency		Information systems support effective planning and control		Information systems linked with competitive strategy

**B. User Profile**

Which type of end-user would characterize yourself?

- \_\_\_\_\_ Command-level end user (can make simple queries and generate reports)
- \_\_\_\_\_ End-user programmer (develops applications)
- \_\_\_\_\_ Functional support specialist (develops applications for end-users within a functional area but is not a DP professional)

**C. User Culture**

Using the 1 to 5 scale provided, please circle the choice which most closely describes the user culture within your department:

1. User Involvement in Application Development
 

1	2	3	4	5
Users let MIS consultants design applications		Users develop some applications		User-developers are self-sufficient
  
2. Computer Literate Support in End User Department
 

1	2	3	4	5
No computer specialists within department		Moderate level of computer support within department		Computer specialists within department help their peers
  
3. Management Commitment to Developing User Support
 

1	2	3	4	5
Users are on their own		Computer literate end users provide user support without formal title		Formal functional support specialists have job titles, clear responsibility
  
4. Training and Development of End Users
 

1	2	3	4	5
No formal program to train or develop end users		Ad hoc efforts to train end users in workshops and using external programs		Formal user certification process with training and education plans support by user management

**D. User Applications**

1. Listed below are several types of applications. If you were to describe your applications, what percentage would fall into each of the following categories:

- \_\_\_\_\_ Operational systems (paperwork processing systems like accounts receivable)
  - \_\_\_\_\_ Query, data extraction, or report generation
  - \_\_\_\_\_ Simple applications requiring logic or computations (subtotals, averages)
  - \_\_\_\_\_ Sophisticated applications requiring complex models for data analysis (operations research, etc.)
  - \_\_\_\_\_ Other: \_\_\_\_\_
- 100%

2. If you were to describe the sources of data for the applications you use, approximately what percentage would fall into each of the following categories:

\_\_\_\_\_ Data extracts from production files  
 \_\_\_\_\_ Data for personal data bases  
 \_\_\_\_\_ Data from other end user systems  
 \_\_\_\_\_ Data from external data bases (e.g. Compuserve)  
 \_\_\_\_\_ Other: \_\_\_\_\_  
 100%

3. What are the characteristics of two of the applications you have developed to support important decision-making tasks?

First Application Name: \_\_\_\_\_ Date: \_\_\_\_\_  
 Information Provided: \_\_\_\_\_  
 \_\_\_\_\_  
 Type of Data Accessed: \_\_\_\_\_  
 Hardware Used: \_\_\_\_\_ Software Used: \_\_\_\_\_

4. System Characteristics and Development Practices:  
Please use separate sheets entitled "Application Characteristics" to describe each application you listed in question 3."

5. For the systems you develop, who is responsible for each of the following systems development functions? Check all that apply.

	MIS	IC Analyst	User	N/A
Conducting the feasibility study	_____	_____	_____	_____
Cost justification	_____	_____	_____	_____
Analysis and design	_____	_____	_____	_____
Programming and testing	_____	_____	_____	_____
Operations	_____	_____	_____	_____
Maintenance	_____	_____	_____	_____

**E. Policies and Procedures for User-Developed Applications**

1. Please indicate whether or not policies and guidelines for each of these areas have been developed.

	No Policies	Guide-lines	Policies developed
Quality assurance	_____	_____	_____
Hardware and software aquisition	_____	_____	_____
Data security and management	_____	_____	_____
Types of applications users can develop	_____	_____	_____
Application development approach	_____	_____	_____

2. Who develops policies and procedures for end user computing in each of the following areas? Check all that apply.

	MIS	IC	User mgmt	N/A
Quality assurance	_____	_____	_____	_____
Hardware/software acquisition	_____	_____	_____	_____
Data security and management	_____	_____	_____	_____
Types of applications-user can develop	_____	_____	_____	_____
Application development approach	_____	_____	_____	_____

2. Who develops policies and procedures for end user computing in each of the following areas? Check all that apply.

	MIS	IC	User mgmt	N/A
Quality assurance	_____	_____	_____	_____
Hardware/software acquisition	_____	_____	_____	_____
Data security and management	_____	_____	_____	_____
Types of applications users can develop	_____	_____	_____	_____
Appropriate development approach	_____	_____	_____	_____

3. Who enforces policies and procedures for end user computing in each of these areas? Check all that apply.

	MIS	IC	User mgmt	N/A
Quality assurance	_____	_____	_____	_____
Hardware/software acquisition	_____	_____	_____	_____
Data security and management	_____	_____	_____	_____
Types of applications users can develop	_____	_____	_____	_____
Appropriate development approach	_____	_____	_____	_____

3. Who enforces policies and procedures for end user computing in each of these areas? Check all that apply.

	MIS	IC	User mgmt	N/A
Quality assurance	_____	_____	_____	_____
Hardware/software acquisition	_____	_____	_____	_____
Data security and management	_____	_____	_____	_____
Types of applications users can develop	_____	_____	_____	_____
Application development approach	_____	_____	_____	_____

2. Who develops policies and procedures for end-user computing in each of the following areas? Check all that apply.

	MIS	IC	User mgmt	N/A
Quality assurance	_____	_____	_____	_____
Hardware/software acquisition	_____	_____	_____	_____
Data security and management	_____	_____	_____	_____
Types of applications users can develop	_____	_____	_____	_____
Application development approach	_____	_____	_____	_____

3. Who enforces policies and procedures for end user computing in each of these areas? Check all that apply.

	MIS	IC	User mgmt	N/A
Quality assurance	_____	_____	_____	_____
Hardware/software acquisition	_____	_____	_____	_____
Data security and management	_____	_____	_____	_____
Types of applications users can develop	_____	_____	_____	_____
Application development approach	_____	_____	_____	_____

4. Is there a policy or procedure for auditing user-developed applications?  
 ( ) Yes ( ) No

If yes, what types of applications are subjected to audit?

---

**F. User Management Awareness**

Using the 1 to 5 scale, assess the awareness of user management regarding the risks and benefits of end-user computing.

1	2	3	4	5
Passive		Aware of risks and benefits; no definite strategy for managing end-user computing		Deliberate strategy to minimize the risks and maximize the rewards

**G. Support and Services for End-User Computing**

For each of the categories of assistance below, use the 1 to 5 scale to rank the importance of each service as well as the actual performance of end-user computing personnel (who work for the MIS function) in providing this service.

Importance	Performance
1 = not important	1 = poor
3 = important	3 = average
5 = most important	5 = excellent

Please circle a response in each of the two columns concerning importance and performance of each of the following services:

Importance						Performance						
Technology												
1	2	3	4	5	N/A	Assessment/research	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Purchasing assistance	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Training (hardware)	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Training (software)	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Support/maintenance	1	2	3	4	5	N/A
Application development												
1	2	3	4	5	N/A	Planning/needs assessment	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Data base design	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Programming	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Documentation help	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Maintenance	1	2	3	4	5	N/A
Communications												
1	2	3	4	5	N/A	Newsletters	1	2	3	4	5	N/A
1	2	3	4	5	N/A	User Groups	1	2	3	4	5	N/A
Consulting help												
1	2	3	4	5	N/A	One-on-one	1	2	3	4	5	N/A
1	2	3	4	5	N/A	Hot-line	1	2	3	4	5	N/A

**Importance**

**Performance**

**Other Services**

1	2	3	4	5	N/A	Data extraction	1	2	3	4	5	N/A
1	2	3	4	5	N/A	<hr/>	1	2	3	4	5	
						(other)						