1980

AN EXAMINATION OF THE INTERACTION BETWEEN TECHNOLOGY, METHODOLOGY AND INFORMATION SYSTEMS: A Tripartite View ·

R. J. Welke
McMaster University

B. R. Konsynski
University of Arizona


• N E W D O C •

AN EXAMINATION OF THE INTERACTION BETWEEN TECHNOLOGY, METHODOLOGY AND INFORMATION SYSTEMS: A Tripartite View

R. J. WELKE

McMaster University
Hamilton, Ontario

B. R. KONSYNSKI

University of Arizona
Tucson, Arizona

INTRODUCTION

In this paper we set out to explore the interaction between technology, methodology and information systems. Our original purpose in doing so was to review the co-evolution of technology and methodology and their interactions. However, in exploring the interactions, it became evident that even within a fixed time frame of methodological and technological advancement there are important interactions which need to be understood if information systems are to be developed which are both "user-successful" and technologically sound. Thus a second purpose of this paper is to highlight what some of these interactions are, how they impact on an information systems outcome, and the need for further research in this area.

To accomplish these somewhat lofty objectives in the short space of this paper, it is necessary to erect a simple (but expandable) framework by which these interactions can be explored. The device chosen is to relate the notion of an information system to two systems: a user system and a (computer-based) data-processing support system (DP system). An information systems methodology is then taken as the means by which one alters, over time, the content of both the user system and the data-processing system so as to introduce into the organization a "new" information system.

CLARIFICATION OF CONCEPTS

USER-SYSTEM

We take, as the target for IS development, a user-system. A user-system we define to be one or more individuals co-operating on the accomplishment of one or more functions in an organizaion. The user-system is said to have criteria by which it implicitly or explicitly measures both the success with which it performs its functions and the satisfaction it derives from doing so. The individuals are said to assume jobs which are in turn comprised of tasks. The assignment of individuals to jobs constitutes the organizational structure of the user-system, the assignment of tasks to jobs constitutes the work structure.

A task, for our purposes, is initially viewed as a black-box transformation which, when triggered, employs external inputs and/or internal knowledge, to produce an output or response. One can, of course, decompose tasks into sub-tasks recursively, assuming that the internal structure of the task can be determined to some level.

Tasks, at a given level of analysis, can be classified along a number of dimensions; we elect two:

1. Decomposible v. Non-decomposable (Definable v. Non-definable)

2. Data-oriented v. Non-data-oriented

By "decomposable" we intend a task which is capable of being decomposed into labelled sub-tasks; i.e., it possesses a determinable structure. At some point in the decomposition, we reach a point at which one can no longer discover sub-tasks (although there may be an underlying "deep structure"); we call this a non-decomposable task. This invokes a sub-criteria: is the task definable or not? This gives rise to the following labels assignable to a task at any point in analysis: decomposable (structurable), non-decomposable-definable (programmable), non-decomposable-non-definable (indeterminate).

By "data-oriented" we mean data as potentially processable by the then existing data-processing system. This might be data used to trigger the task, to supply external input to it, to alter its

knowledge base, and/or to accept its output.

These are not objective criteria. One person (analyst) may discover task structure whereas another may not. Similarly, one observer may "see" data usage related to a task whereas another may not. This, in a broad sense, leads us to the type of interaction between technology, methodology and information systems which we wish to identify in this paper.

DATA-PROCESSING SYSTEM

A data processing system is in many respects similar to the way in which the user system was described. It has a mission, criteria, jobs, and tasks. However, there are important, fundamental differences. First, for purposes of our exposition, the tasks are performed by a computer, not people. The resulting work structure is organized into modules, routines, programs, and systems of software. The organizational structure of processors assigned to jobs constitues the configuration or "architecture" of the hardware. The criteria which apply to the DP mission are typically different from those applied to the user system and do not contain the inherent satisfaction criteria of the user-system. This tasks performed by the DP-system are, at the elementary level, well-defined and completely data-oriented.

INFORMATION SYSTEMS DEVELOPMENT METHODOLOGY

Employing the above concepts of a user-system and a data-processing system, we define the process of information systems development to be the alteration of the user system to permit the transfer of data processing tasks to the DP-system. Good IS development is when the transference of such tasks results in an acceptable (from a satisfaction criteria sense) and improved (from a mission sense) user system change as well as an acceptable addition to the DP-system (from a DP-system criteria sense). To guide the developer(s) in this change process to the user and DP-system, a "methodology" is employed. Actually, many methodologies may be used in the process, but to simplify discussion, we will refer to them as if they are part of one overall methodology. This "overall methodology" may in turn have different sub-parts dealing with different aspects of the development/ change process.

A methodology, in our view, consists of three separable aspects:

1. How we elect to see the object system to be changed (the DP-system and the user-system).
2. How we elect to go about the act of developing itself.
3. The skills and techniques employed.

The aspect of seeing (or, more correctly, perceiving) we will refer to as the object system context. The concept is quite simple. A developer, faced with an object system to be developed, identifies certain objects, their properties, and the relations between these objects; i.e., he/she filters the system being observed or conceived so that it is seen to be made up of these specific subjects, properties, and relations. A "canned" methodology enforces a particular world view on the developer using it by pre-selecting the context to be employed. There are many such contexts. Some apply solely to the user-system, some only to the DP-system, and some to both. Each highlights different features of the system being developed.

Regarding the aspect of how we go about the act of developing itself, we refer to this as the development approach. One facet of the development approach is the sequencing of activities. A common example here is the Systems Life Cycle, although there are many alternatives (e.g., PSC). Another facet is the mode of participation, i.e., who does what and what manner. Here things can vary from complete autocracy to complete participation. There are many other development approach facets, as well, combining to form a variety of development approaches.

Operating between the selected context(s) and the development approach(es) are the skills and techniques which are employed by the developers to "fill-in" the contexts for an existing or proposed object system. These vary from the more traditional skills of interviewing, observation, and communication, to the more refined techniques of elicitaion, and data and task structuring, to name but a few categories of techniques.

The assemblage of selected contexts, development approaches, and skills and techniques into a semi-coherent whole constitutes what we will call an IS development methodology. We do not claim that such a holistic methodology exists under one name; "methodologies" such as structured analysis and design (e.g., SADT), BSP, ETHICS, ISAC, etc. are best viewed as partial methodologies

representing selected contexts, selected development approaches and supporting selected skills and techniques directed at specific aspects of the system to be changed (DP and/or User).

Having outlined the "triangle" of: user-system, DP-system, and IS development methodology, we will now take up each of these components in greater detail, with respect to their advances an the interactions between these advances and the other members of the triangle.

THE DP-SYSTEM

It is impossible, in the space of a few pages, to highlight all of the important technological advances which have taken place since the first commercial application of computers to business. As such we will focus on the following:

- Data management and DBMS
- Centralization and Distribution of Data and Processing
- Data Integrity and Internal Controls
- Technological Advances in Cost/ Capacity
- Applications Development Software

The interactions of these changes are to be seen in both the methodologies used and in the resulting changes (type and form) to the user system. They are also reflected in changes to the DP system criteria. Regarding methodologies, changes in the above areas have resulted in altered contexts and new techniques for development. Tasks which five years ago were overlooked as not amenable to formal data processing are being seen in a different light. Relations between data, once seen in terms of fields records and files, are now being seen in terms of entities and relations, documents and non-documents, etc.

Changes have also taken place in the user system in terms of expectations and new fears. Users have come to expect flexibility and rapid response to needed changes. Operationally they expect rapid response time, little down-time, hassle-free recovery from system crashes, the ability to formulate ad hoc inquiries, etc. Fears of job displacement have been augmented by fears of poor job re-design, privacy of personal data, and such things as the radiation effects of VDT's.

These are but some of the interactions between technology and methodology and the user-systems. We will now focus on some of the more specific areas mentioned above.

Data Management

In the past thirty years, major changes have taken place in the manner in which organizations manage the data resource. Radical re-organizations are taking place in organizations as a result of trends toward data sharing and centralization of the control over information. The evolution of Data Base Management Systems (DBMS) has made a significant impact on the ways in which management expects to control the information resource.

With the integration of previously heterogeneous data files, the availability of information has significantly increased. The users of individual data items have their information available along with on-line access to other informaion that was previously unavailable, or difficult to acquire. The integration and standardization of interfaces has expanded the community of interest. New users are now able to access data in a way that way previously too complex to accomodate. Added to this are advances in data integrity, control, and timeliness.

The control of the data management function is no longer the exclusive domain of the collecting agency. Many organizations have special Information Resource Management functions whose responsibility is the effective organization and delivery of information to the operating departments of the organization. Indeed, these managers of the information resource frequently use data management techniques themselves to assist them in their managerial tasks (e.g. Data Dictionaries/Directories).

We use the term data base to refer to the collection of data that is used by the organization and its application processes. Thus, the technology offered by a Data Base Management System (DBMS) is a tool for management of the organization's database or data bases. DBMS technology has offered a new data independence, the independence of the logical and physical data organizaions.

The data base management concept involves the use of computer-aided tools to manage the data resource, including such functions as:

- Support for the growing complexity of data relationships
- Facilitate general, uncommon queries to

the resulting data resource

- Respond to ad-hoc questions in a timely and economical fashion

- Promote the sharing of the data resource, and increase accessibility

- Reduce the development costs of the applications that use the data resource

- Improve the quality of the integrity controls on data

- Reduce data redundancy and its related update problems

This is the promise of DBMS, a promise made possible by advancing technology and theory applied to data management. To realize this promise, however, new contexts and techniques were required, both for the development of the user system and for the DP-system. Many of these are now available. However, they have not been widely integrated into the methodologies employed by the IS developers. If one cannot "see" a DBMS in the user system, it is difficult to design for one. Not surprisingly, it is much easier to see a DBMS structure in existing computerized files, as many of the tools and techniques are available to aid conversion of existing file management systems, rarher than the development new systems. This has resulted in a time lag between the implementation of DBMS as a technological construct and the broader user-system view of data as a resource and the concommitant data-managed designs. Only recently have such concepts as the entity-relationship model been put forward to enable the user system developer to perceive the data aspect of a user-system in a manner consistent with the technologically based DBMS -- a lag of nearly ten years!

Regarding the interaction between this technological advance and the effects on the user system, these are only now being studied. Issues such as the shift in knowledge-based power and the resulting impact on organizational responsiveness are just beginning to be felt. More rapid response to ad-hoc inquiries, while on the surface appearing desirable, may in fact be detrimental to organizational performance if used in a control setting.

As more of this becomes known, we can expect further changes in the contexts applied to user-system development, to examine new relationships and properties in an attempt to guard against dysfunctional side-effects of introducing the DBMS view of data management into an ill-suited or unprepared user-system.

CENTRALIZAION AND DISTRIBUTION

Technology has offered management greatly increased opportunities for the distribution of information and the decentralization of data collection, while retaining centralized control. The result has been an increase in the managers' sphere of control or manageability, while at the same time, off-loading the more mundane functions to remote sites.

We can trace the technological evolution and relate the impact of the various stages to the conduct of management in our organizations. Initially, all functions were centralized in the data processing centers. This situation was necessary due to the economies of scale and the availability of personnel, as well as the status of data communications. Information collected at remote sites had to be transported in a source document form to the computing facility. All processing on data took place at the central site; only completed documents were delivered to users, with little or no input from the user during the processing.

With the advent of time-sharing and other facilities for remote communication, an interest developed in "real-time" transaction processing. The result was a tremendous increase in the community of interest in DP. The capabilities of the computer were made available to large segments of the organization that previously had no interest in DP. Thus began the wholesale realizaion of the man-computer concept.

Next, remote storage and inter-node interchange technologies enabled the user to gain increasing control over the data in his/her possession. With these innovations, the concepts of inter-user communication and infomration-sharing are realized. The evolution of these technologies and associated concepts lead to the concept network architectures building on the related technologies of processor hierarchies and network tasking. With these new architectures, decentralized, evolutionary design, and implementation strategies became technologically feasible.

We are now at the point where the full distribution of processing and data management functions is possible. The current technology facilitates the full distribution of data, directories, programs, and processes, in addition to the distribution of control functions. The technology allows for the effective use of micro-processor capabilities in networking small processors to create

sophisticated systems. Distributed Data Processing, as it is called, gives the control function to the network itself.

The promise of DDP is to allow the user to work at his/her location, with data of the users, choosing, in a form/structure of the users, choosing, presented in a way the user finds most useful, without the necessity for managing the data control functions necessary to accomplish this. To realize this promise, however, requires that both the systems development methodology and the user system be brought into synchronization with the technological promise. As is traditionally the case, both methodology and user-system preparedness are lagging. For example, do we have adequate contexts for the representation and appreciation of network versus hierarchical control as it will apply to the user system? Do we understand the social, political, and task consequences of such a shift in task distribution? What design heuristics and/or selection criteria are available to the developer to select between the different processing architectures now available? Or, is this strictly a physical implementation rather than logical design considertion?

DATA INTEGRITY AND INTERNAL CONTROL

Data must be consistent with propositions and assertions regarding values, relationships, and use of data. With the proper application of current technology, the definition and implementation of controls at all stages of the "data life cycle" are possible, resulting in improved data integrity, which in turn determines the utility of data to the manager (i.e., recency, accuracy, consistency, and precision).

During the "data life cycle," many processing stages exist in which data integrity problem can occur. The basic stages in this cycle can be characterized as:

1. Measurement and classification
2. Transcription
3. Validation
4. Storage
5. Processing and Updating
6. Recovery

Technology, coupled with appropriate design techniques, offers many solutions to the integrity concerns at all stages of the cycle. This is taken up below.

With data measurement, problems relate to the collection procedures, the measurement scheme employed, and completeness. Technological innovations in data measurement and data entry capabilities have made the notion of source data capture a reality, with the related advantages of source data checking, immediate error-feedback, intelligent monitoring of input sequencing through on-screen prompting, the potential elimination or reduction of error-prone steps (e.g., transcription), and increased recency of data. The result has been substantial improvements in the measurement, transcription, and validation areas of integrity. Non-technological advances, such as our understanding of measurement as it relates to what we are observing and to how the measurements will be employed, have been slower in coming and in being applied.

The problems of data storage involve the inital creation and subsequent updating of files and databases. The basic organization of data in storage has an obvious effect on the procedures for maintaining data integrity. Internal control functions and data management systems are an opportunity for the formalization of control requirements in this area. The independence of logical and physical data structures now available as a feature in many DBMS's permits controlled access, controlled update, and extensible definitions of data without the need for error-inducing redundancy.

Related problems for privacy, simultaneous access/update, and system crashes have been both exacerbated and reduced by the introduction of new technology. The privacy issue is being countered with techniques ranging from privacy locks on data to encryption techniques. Simultaneous use has spawned new sub-disciplines within DBMS which are developing a range of techniques to deal with these problems. Hardware and software failures, resulting in data contamination, have lead to new approaches to re-start and recovery procedures which make use of a variety of techniques including incremental dumps, dual recording, transaction logging, and flagging. Notably, all fo these functions have been implemented within the hardware/software technology itself, making their existence transparent to most users.

Again, on the user-system side of life, progress has been much slower. A simple example of this is the area of EDP auditing. As more and more of the old paper audit trail is umbrellaed by technology, the problems of the internal and external auditor compound. With very

few exceptions, the audit profession has not kept pace with the advance of technology, leading to an increasing gap between the ability to "attest" to the integrity of (financial) data, and the data actually contained in the system.

TECHNOLOGICAL ADVANCES IN COST/CAPACITY

One of the most sustained historical trends in the area of hardware is the on-going gains in raw computational capacity for a given amount of money expended. This has been matched by similar increases in cost/performance in the hardware peripheral areas. This has had several effects. For larger systems, a greater proportion of the "managerial" tasks have been subsumed by the system, resulting in greater function, flexibility, and scope in terms of what constitutes a data-processing task. Color computer graphics, once the exclusive province of large universities and high-technology application areas, are now changing the way in which senior executives see their data. Document storage and processing, once relegated to libraries and large publishers, are now the building blocks of word-processing and office automation for small offices. In short, gains in the cost/capacity curve are constantly redefining which tasks are candidates for transfer from the user-system to the DP-systems.

A second effect of the advancing cost/capacity gains is the availability of computing to firms who only a decade ago could not have afforded a line printer let alone the computer system to drive it. Now, a whole new level of organizations are taking on computer-based data processing on a scale unprecendented by the past three decades of data processing, but with some significant and potentially dangerous differences. The systems being implemented are generally on-line applications, skipping that intermediate organizational learning step called batch-processing. They are being implemented without the benefit of individuals seasoned in systems development and in the problems of implementation. The computer systems themselves, while sophisticated from a functional point-of-view, lack the basic data integrity functions which larger systems have, out of necessity, implemented. The lessons learned by the larger organizations in evolving a DP-system are being re-learned from scratch by the smaller ones; everything from stream-of-conscious coding (vs. structured design) to wholesale changes in the user system without an appreciation for the dysfunctional consequences such changes frequently bring.

In both cases we once again observe the situation where technology is outstripping the ability of development methodologies and user systems to absorb the advances. New contexts are required to broaden the developers ability to "see" new DP tasks in an existing organization along with new development approaches and techniques to successfully integrate them. For smaller organizations, new development approaches must be developed which are consistent with the size of the organizations and the resources available for development. "Quick and dirty" techniques combined with the more efficient application of existing tools are badly needed for this level of organizational implementation. Unfortunately, the main focus of information systems development reserch is still directed at the large, more mature organizations.

APPLICATIONS DEVELOPMENT SOFTWARE

Paralleling the advances in cost/capacity has been the advances in software languages which permit the transfer of user-system tasks to the DP-system. From the machine languages of the fifties, we have now entered the age of (high-level) non-procedural languages and applications system generators. There are a number of implications of this technological advance in terms of information systems development. At the data processing level are the advantages of development efficiency and flexibility. This applies both to initial systems development as well as the on-going maintenance of the resulting system. Prototyping, a much used concept in the development of physical systems, becomes increasingly practicable, as does the frequently discussed notion of users developing their own systems without the need for an intervening programmer/analyst. More consistent systems quality, integrity, and the ability to deliver on-time and in-budget are other potential advantages. Such is the promise of these advances.

To realize these benefits, changes are necessary in the related areas of information systems development methodologies, user system attitudes, and DP-system criteria. Application systems generators have imbedded in them a view of how the resulting information system should look, at a level much higher than the traditional function/module level. To employ such software requires the developer to see the target user system in a much coarser fashion than the traditional bottom-up analysis techniques easily permit. Some generators have an implicit data-centered outlook, working

from an assumed centrally managed database system. Others are more process/task oriented, maintaining the older traditions of separate transaction and master files. Still others are data entry/report-query input output oriented. These are distinctly different development outlooks just as centralized vs. distributed are different approaches to processing system architectures. One should have tools and techniques to assess these differences in the context of a particular development. More importantly, the developer must appreciate that these are different development approaches which have long-term conswquences for both the user system and the DP-system.

The availability of applications generators and non-procedural languages has the potential of transforming the concept of an "information system as a product" to that of an on-going service, as seen by the users. The "lion's share" of information systems are at present seen by users as products which are supplied and maintained by the DP-system. A system is "purchased" from the DP group and "sent in" for repairs and modifications as required. When the old system becomes unwieldy, it is scrapped and a new product is designed and delivered (the product/system life cycle). With the ability to let the users devise and adapt their own system, DP becomes a service (like, say, electricity) to which they supply their own value-added software to derive function. This represents a major shift in user thinking towards DP, one which few users are prepared for, and one might add, few DP groups are prepared for as well.

SUMMARY

As previously stated, this exposition is not intended to be an exhaustive discussion of the effect of technological advances on the changing nature of information systems. Rather, we sought to characterize some of the advances, indicating the cross-interactions between technology and related development methodologies and the recipient user system. While there continues to be debate in some circles on the extent (if any) to which technology (out)paces development and the user-systems ability to assimilate these changes, we hope that the preceding discussion gives evidence that technological advances do indeed continue to outstrip the advances being made in development methodologies and the user-systems' capacity to absorb effectively such advances. As such, we hold that an awareness of technological developments by information developers and researchers is an important reference discipline for Information Systems.

THE USER-SYSTEM

Just as there have been important advances on the technological side of the information systems development "triangle", there have also been advances in our thinking regarding the user system, which have had and will continue to have as important an impact on the development of information systems as the technological advances discussed above. However, because they are not conveniently packaged into software and hardware products, they are not as easily defined and discussed.

Again, we are forced to highlight rather than exhaust the imaginary list of candidates for discussion. We choose the following as exemplars:

- The user as part of the system
- The broadening of the scope of "DP-able" tasks
- The role of the users in the development of the system
- The redefinition of the systems developer as change agent
- The shifting of criteria defining information systems success

THE USER AS PART OF THE SYSTEM

The role of the user information systems development has changed almost as dramatically as the shifts in technology discussed in the preceding section. From an object system context point-of-view, the generic user has been elevated from the equivalent of a pliable task execution mechanism to an organic entity in its own right which is associated with a number of tasks through a set of position/task relationships. From a development approach point-of view, the user has gone from an interviewable source of information regarding the existing system and recipient of the results of the design, to that of becoming an active member of the development group itself. In this section we examine briefly the former transistion regarding the user vis-a-vis the object system.

The shift from a "mechanistic" to an "organic" view of the system under examination and development has been slow to emerge. Early information systems thinking concentrated solely on the tasks being performed in the system. Who performed these tasks was of little concern except to identify who to interview during the analysis stage of systems development. The main focus of inquiry was to identify "programmable"

tasks and to then program them (i.e., transfer them from the user-system to the DP-system). The resulting physical implementation of the redefined job-task (work) structure was then "managed" through organizational redefinition and training. Such implementations did not always lead to satisfactory results, leading to the notion of user system design failures as expessed in terms of economic and technical goals.

The types of failures with respect to the user system can be classified in a number of ways. One useful partitioning is between captive users and discretionary users. Captive users, simply defined, are those who must use the resulting system in order to perform their organizationally assigned work. Discretionalry users are those who have alternatives; as such, if they don't "like" the resulting system they simply don't use it. The captive users were the first to be dealt with, at least in terms of academic examination. Mumford, Hoos, and others began noticing at an early date the consequences that mechanistic systems development was having on the captive users involved in the system being changed. This lead to the emergence of what is popularly known as socio-technical systems design approaches related to information systems development, whereby the user is seen, not merely as an embodiment of tasks, but also as a person who has various needs to be fulfilled by or through the work performed. With this broadened view of the user system, not only are inter-task relations of importance, but the relation between tasks and jobs as well. Furthermore, the criteria of what decides an acceptable system is broadened to include job satisfaction criteria which are in turn related to the job-task (work) structure.

Discretionary users, on the other hand, were slower to be incorporated into the definition of the user system. Typically, the discretionary user is allowed to remain so because of his/her organizational position and/or the "non-programmable" nature of the work the person performs. Initially, the discretionary user was sought out and used in much the same way as his/her captive counterpart; i.e., the user was interviewed in terms of information he/she required from the system. The discretionary user differed in that he/she was treated as a "black box". Here to, failures were noted, e.g., reports were not used. This lead, initially, to the evolution of more sophisticated techniques for eliciting requirements, called MIRS or Management Information Requirements Analysis techniques. Several inventories of such techniques have been compiled (cf. Bariff, Taggart). At most, they required a separate object system view -- that of the discretionary users surrounding system, in order to develop protocols for information requirement elicitation. Alternatively, the user was "investigated" according to his/her cognitive style, in order to better understand the form of both the information and its delivery that would be acceptable. The discretionary user, however, remained at the periphery of the system as a "sink" rather than an integral part of it.

The inclusion of the discretionary user into the system under development came with the emergence of Decision Support Systems (DSS's). From its "humble" beginnings as an exploration of the potential use by managers of time-sharing (a technological advance), DSS has emerged as the umbrella for most of the work directed at allowing discretionary users having computer supportable judgmental tasks, to be considered as an active part of the user/DP-system under development.

The above discussion of captive/discretionary users tends to highlight the end-points in user-inclusiveness. An important intermediate stage was the identification and pursuit of man-machine interactions. This was prompted by the emergence of on-line systems in which the user "squared-off" with a terminal. Initially, such work was directed at the military systems and drew from previous man-machine couplings (e.g. radar systems). However, increased attention was given to "user-friendly" interfacing, "error-reducing" input sequencing, and the evaluation of alternative display formats. This work, at minimum, recognizes the user/system interface as part of the user system under development, and as such implicitly recognizes the user as part of the system as regards general properties that a user possesses that will result in a more or less acceptable interface. Whether this recognition is incorporated into general design heuristics, or alternatively accepts the notion of customized interfaces for particular individuals, determines whether the user has become "part of" the system under design.

THE BROADENING OF THE SCORE OF "DP-ABLE" TASKS

In the introductory section of this paper, we suggested that one important relationship between the user system and the DP system was the ability to see DP-able tasks in the user system. What this means is that the "domain" of the user object system, as observed through an information systems development context,

is constantly changing because our definition of what should be observed and represented in terms of tasks for possible change. Again, historically, the perception of user systems was limited to identifying programmable tasks and formal(izable) records as defined by what was economically processable by the current technology. As this set expanded, thanks to both the cost/capacity advances of technology and our ability to make use of these advances (as demonstrated through "state-of-the-art" applications), the IS development perception of the object system changed. Tasks and data flows initially excluded from representation for reasons of volume, fast accessability, task complexity, and other processing requirements are now included. As such, the scope of examination and potential change is rapidly broadening, not only in terms of the types of tasks and relationships that are now being seen within a given object system, but the types of user systems that are being examined.

A current example of this is umbrellaed under the term "office automation," i.e., the penetration of the office work environment by DP-systems. What has permitted this to occur, among other things, is our ability to "see" in the office environment (read: user object system context), DP-able tasks which were hitherto not seen as DP-able tasks. Among these are: text manipulation and storage (in distinction from fixed record manipulation and storage), known as word processing; the transport of addressable documents (as opposed to point-to-point movement of data), known as electronic mail; the filing and retrieval of documents (as opposed to fixed records); and numerous other record keeping activities previously seen as ad hoc and/or not worthy of computer support in and of themselves (e.g., appointments, petty cash, name-and-address files and even the checking for spelling errors).

DSS also comes under this heading as an extension of DP-able tasks into the middle and upper echelons of management. Tasks historically partitioned into either programmable or non-programmable in their entirety are now being looked at as a continuum in terms of degree of structure and DP-support. Ad hoc task needs are no longer outside of the domain of DP-able tasks, thanks to flexible inquiry and reporting capabilities, as well as increasingly higher level languages which the user can directly employ. With these tools, the concept of DP itself is changing (as previously discussed) from one of subsuming tasks and producing products, to that of other than the way in which it is stored is no longer a barrier to the use of the data, thanks to sub-schema or relational capabilities in database management systems.

In short, the scope of the perceived user system is rapidly being extended due to the advances in technology and the creativity of the developers of the "leading-edge" applications which make use of this technology. In this regard, the role of the universities in developing these leading-edge applications must be given greater acknowledgement and support than has been the case.

It is difficult to predict the next broadening of scope which will take place. Theoretically, the processing, storage, and interpretation of graphical information will become increasing DP-able. So also,will voice communication. In-roads into other sensory and psycho-social areas are also probable, constantly extending the scope of the user system perception as embraced by IS development.

The question that must be asked, however, is: is our ability to represent and anticipate the results of such changes keeping up with our ability to see and make such changes; are our object system contexts adequtely representing and pointing out to the developers of such systems the "by-product" consequences of making such changes; and is there a "snail-darter" waiting for us? The answers are most probably no, no, and yes, respectively.

THE USER AS A DEVELOPER

As mentioned above, the "user" has been looked upon in two ways in the emerging discipline of information systems: in relation to the object system under development as a potential "organic" entity, and in relation to the development activity itself. These are complimentary relationships, but one does not necessitate the other. In fact, in much of the literature, these are discussed as alternatives. The user as a developer has had a varied history. Initially, systems were developed exclusively by DP-systems analysts/ designers, with little or no involvement of the users except as interviewees. Over time, it became important (again due to the recognition of user-system implementation failures) to somehow include the user in the on-going activity of design and implementation. A number of approaches were tried: user representation on the development team and, structured walk-throughs of existing and proposed system documentation. For many organizations these became "hostage-taking" and "structured-walkover"

incidents respectively. Simply "involving" users was not enough. This lead to a re-examination of the objectives in user-involvement. Some of these were: developing a consensus amongst the users that this was or would be an acceptable system, allowing the users to perform a value-added design role (rather than an approval/dis-approval control role) and transfering to the users the "mental" ownership of the results of development.

This has lead to a number of new roles for the user in the development of information systems and to the redefinition of the developmnt approach itself. These will be taken up in the last section on Methodology. However, it should be noted here that many of these "innovations" resulted from the inability of the object system contexts being used to cope with the organic dimensions of the user system. As more becomes known about the results of user-inclusion on the development team, we can expect greater changes in the representation of the user in the object system context itself.

THE DEVELOPER AS A CHANGE AGENT

The ever broadening perception of the user system has brought with it a concomitant change in the role of the developer. Once viewed more narrowly as the "engineer" of a data-processing system (e.g., "systemeer"), the broadening definition of what constitutes a user system has necessitated fundamental revisions in the outlook of the developer role. The specific qualities and alteration of activities of the developer will be taken up in the final methodology section. In this section we focus on the ramifications for the user system context.

A "key" concept is that information technology is an intervention permitting an "unfreezing" of the existing organizational system. The developer is then seen as the agent of change. This outlook can be confined to merely the DP-able tasks, requiring nothing new in terms of the perception of the organization. However, it is intended that such a view should give rise to an expanded appreciation of the "secondary" changes wrought by a change in the re-arrangement of DP-able tasks, and making them a primary consideration in the redesign of the system. This does necessitate a revision and/or extenstion of the perceived user system.

Some of these extensions have already been suggested above: perception of the altered work structure arising from socio-technical considerations; perception of the individuals as entities prossessing cognitive styles, work attitudes, and preferences; the interface between man and machine as a representable and customizable sub-system. Other aspects of the user system are also being brought into the domain of the ISD perception of the user system. These aspects include: the organizational structure and the formal and informal work group sub-systems, as well as their attendant properties of consensus formation, authority and responsibility, and power. In short, there is an increasing overlap between what was once the domain of "organizational development" (OD) and the expanding role of information systems development; mainly because ISD and its practicioners are, for the moment, organizationally legitimized change agents who do change the organization either intendedly or as an un-controlled by-product of the more narrow perception of their roles.

Again, we can ask the question whether or not the methodologies in general and the object system contexts in particular have kept pace with these developments; and again the answer is generally no. We have piecemeal models borrowed from OD and other areas, but they have not been integrated as yet into an ISD methodology, except as add-ons. As a result we are again in danger of the "Dearden effect" of taking on more than we can mentally handle. However, the necessity for taking on the OD aspects should be clear -- we have been indirectly practicing OD since data processing began intervening in the user systems; we have simply ignored it and its consequences to the organization.

USER SYSTEM CRITERIA WITH RESPECT TO IS DEVELOPMENT

All design and development proceeds by explicit and implicit criteria. Problems and opportunities are identified based upon evaluation criteria, design choices and alternative selection employ criteria, and the domain of the systems being percieved are guided by the criteria in place for deelopment. As such all of the above user-oriented changes can be reflected in the changing criteria of IS development, as it applies to the user system.

Historically, the emphasis was on the original two categories of feasibility: economic and technical. For most purposes, economic meant cost savings which translated to improved efficiency of the DP-able operations. Technical criteria reflected the interests of the DP-system management. Given the high fixed cost of computational power and the newness of the technology, these were understandable criteria. However, as the

fixed cost dropped (and/or was spread over a wider base), and the "craft" of DP-systems development became more scientific, the dominance of these criteria became suspect. Criteria reflecting the user-systems needs beyond that of improved efficiency and better control (through improved reporting), were needed. Several categories of criteria emerged: for captive users -- job satisfaction, for discretionary users -- actual use. This lead to the study of relationships between ISD alternaives and job satisfaction and use. An interesting result was that "success" in terms of satisfaction and use was as much a function of how (it was done) as it was of what (were the results).

Other criteria have also emerged, reflecting the dynamic aspects of organizations and the imbeded information systems: recency of data, timelines of access, and flexibility to modify and extend the initial system. Now, we are beginning to see the entire array of organizational unit goals beginning to be aligned against potential, planned, and existing information systems. Whether this is reactively done through "anti-implementation" strategies or proactively done through top-down organizational/information system planning exercises, it is being done.

Are our methodologies keeping pace with these changes in attitudes towards IS development? Do the methodologies reflect these broader sets of criteria? In part, they do. Partial methodologies exist to attempt to cope with IS planning and work-system/DP-system design (socio-technical systems development). However, the integration of this work has only begun, and the cognitive limits of the IS developer to see these broader OD interactions are presently being stretched beyond reasonable limits.

SUMMARY

The user system has had an almost benign existence in the historical pattern of IS development; it was something one did something for or to rather than being a system to be perceived and understood in its own right. This attitude is changing. A two-world view of data-processing systems and user systems is being accepted with the consequent reflection in both the much broadened perception of the user system and the acknowledgement of user system criteria which are frequently at odds with DP-system criteria. To some this might suggest a "deadly-enemy" confrontation between DP and the users, to others an evolving extension of the domain of development of ISD. Either way, there are a number of important interactions that must be understood and managed between the two systems, interactions which are only now coming to the surface even though they have been there all the time.

METHODOLOGIES

A method is defined to be: a way of doing things, especially in accordance with a definite plan; a methodology is said to be: the logical principles underlying the organizaion of the special sciences and the conduct of scientific enquiry. In this section we examine the interaction between the user and DP-system through intervening IS development methods. Our perspective is that of the underlying methodology.

In the introductory section we introduced the partitioning of a method (and its underlying logic) into the object system contexts employed and the development approach taken. While a method inevitably contains both components, their separation permits a more incisive and critical look into the emergence of methods in concert with DP technological developments and a broadening awareness of the user system. Our agenda for this section, then, is to look at the emerging object system contexts, the revisions and extension to the development approach, the interactions between these to "aspects" of methodology, and the activity of information systems development as an object system in its own right, which is in turn amenable to technological and "user" system improvements. First, however, we will examine briefly the philosophy of information systems development.

INFORMATION SYSTEMS DEVELOPMENT OUTLOOKS

The development of information systems is assumed to be motivated by some underlying development "philosophy." But what is that philosophy? This is a very broad question, only two aspects of which we will pursue here: what are the assumed values associated with engaging in information systems development, and who or what are the guarantors of the results? This in turn leads to a number of other interesting "philosophical" topics, e.g., what is the logic of our "modes of inquiry"? what are our "ethics"? etc. A related question is: is our discipline ready for such an examination?

The reason for critically examining the value orientation of our discipline comes from the simple observation that we are actively changing the environments with which we deal. In doing so what "ideal" are we moving the organization and

its environment towards? Governments have constitutions (or their equivalent) which give broad direction to the policies adopted and the actions taken; corporations have charters to allegedly do the same thing. What is our charter? Some would answer that it is the DP-systems charter; others would answer that it is the organization's charter. In truth, the question of values held has not been broadly addressed except at the evaluation criteria level.

The need for doing so should be evident, however. Just as organizations and institutions are undertaking critical self-examination in this area using such techniques as stakeholder analysis and strategic assumption analysis, it is perhaps time that we stand back from what it is that we are doing and inquire into our underlying value system.

There have been a number of value systems put forward in the systems context. Kling, in his examination of EFTS (electronic funds transfer systems) puts forward the models of:

- Private Enterprise: Profitability of the firms providing and using the systems
- Statist: The strength and efficiency of government institutions are the highest goals
- Libertarian: Constitutional civil liberties are the highest goals
- Neopopulist: Understanding and consideration of the common man is the highest goals
- Systems: The system itself should be technically well-organized efficient, reliable, and aesthetically pleasing

Klein, in a recent paper, suggests that the above "design ideals" are not derived from a well-reasoned philosophical base. He draws from the field of ethics to criticize the prevailing (default) design ideal of cost/benefit analysis as a form of utilitarianism, employing the arguments of Rawls (1971). He then examines quality of working life (QWL) as an alternative ideology which begins to focus attention on the hidden social and psychological costs of (utilitarian) managerial action in general and the continued automation through computers in particular. However, the adoption of QWL principles is not without its difficulties as well. As an alternative, the author puts forward the concept of "socio-responsiveness" whose primary values are those of consensus and a community of shared beliefs.

It is not possible to explore fully the concept of socio-responsiveness here in terms of its implications for IS development and its diagnostic power regarding existing design ideals. Nor is it possible to discuss other possible value systems (e.g., Marxian). What is important is that such value systems have an important effect on the construction and utilization of a method. As such, a more open debate is needed on them than is presently afforded by the literature.

The "guarantors" of our results also bears discussion. Churchman, in his book on The Design of Inquiring Systems, points out that as developers of information systems, we are in effect treading in the philosophical paths of a number of great philosophers and should look over our shoulder to gain insights from the (still raging) debate on how we know, and know that we know. We can even map out the progress of our discipline as a direct parallel to the progress made over the past centuries in the philosphy of science. Starting with a Liebnetzian view that the guarantor resided in the models we constructed and used in a normative fashion, to the Lockean community of selected users and analysts forming the design team, to the Kantian-like use of prototypes to achieve truth in inquiry, to the now fashionable dialectic development approach of Hegel, and even to the evolutive strategies which have their parallel in Popper's scientific revolutions. The preceding is an admittedly loose application of the notion of guarantor; however, it should suggest that it underscores a historically valid and widely recognized problem regarding inquiry which has direct implications for the validity of out own IS development actibities.

OBJECT SYSTEM CONTEXTS

Throughout this paper we have constantly referred to the object system context associated with, or affected by, a particular user system or DP-system change. We now expand on this concept both as a concept and as a way of highlighting some of the current thinking on information system development methods.

The importance of object system contexts as a separable concept is that it permits discussion about how both users and developers "see" a system as a reflection of their own, often divergent "world-views." The contextual frame is a subset of one's "world view." It is generated via a priority process wherein the individual explicitly or implicitly identifies a set of objects and relationships in the world view. This set can be heavily influenced by a the set implicit in the adopted "method" being used for development. Thus one can view a specific method as providing one set of

objects and relations. Given that there are a number of such contexts amongst the many competing alternative mehtodologies, there is, in principle, the choice between the contexts which could be used.

Given that there is such a choice (a point we will demonstrate shortly), one must suggest a basis for adopting one over the other. The basis suggested is that of the evaluation criteria to be applied to the results (which in turn should be derived from the value system adopted for IS development). In theory, these criteria should be known in advance; in practice they may be quite vague, offering little in the way of precise measures. However, even as ill-defined concepts (such as sytem flexibility), these criteria are integrated into the contextual frame of reference to establish an internal object system image to which the final results are "targeted." Initially, this frame consists of a number of internal images residing in the minds of the developers. An important function of any design process is to generate external representation of these images to permit communication amongst the individuals involved. These we can call object system "models."

To summarize, at the initiation of an information systems development effort, there are a number of "actors," each possessing an underlying world-view which must be focussed and refined with respect to a particular object system. Object system contexts, which highlight particular objects and relations (and their associated properties) are either adopted or internally generated by the actors to enable them to percieve what is to be developed. We postulate that this selection is governed by the underlying, evolving criteria to be applied to the developed system. This process enables each actor to "see" the system but not necessarily to communicate with the other actors. To communicate (and to remember) an externalized image is needed -- the object system model.

Existing methods can now be seen to possess specific underlying model frameworks; they identify and generically name the objects and relations to which they attend. This is not to imply that there is only one type of model; some methods identify a variety of objects and relations, considerably expanding their imaging power. Nevertheless, they all focus development on some set of objects, relations and properties and typically presume some underlying criteria which their properties reflect. Some examples are now in order.

Yourdon's "Structured Analysis and Design" method provides one current example. It is best discussed in terms of the generic models it employs. The first is the "data flow diagram" which permits a M:M data flow relation between objects called activities, files, and externals. This provides an overall topology of the system under study. Subsidiary models in the form of data dictionaries and structured English provide specific projections out of this model to more detail regarding the data-flows/files and the activities respectively. The Bachman "data structure diagram" is used to represent the "entity-relationship" between data records. "Structure Diagrams" are used to represent the data passing and control relationships between processing modules collectively directed at automating an isolatable function/requirement extracted from the data flow diagram model during the logical design stage.

Thus, the Yourdon and similar approaches put forward a limited number of models and their underlying objects and relations in the form of a method which an information systems developer is asked to use. The question is: are these objects and relations sufficient for IS development? The answer, as we've suggested above, depends upon the criteria being used. For example, if the maintenance (or improvement) of job satisfaction is a criterion, does this method permit its inclusion? The answer is easily seen to be no; job satisfaction is a property of either an individual or inherent in a particular job -- neither are given explicity representation this method. What of the effects on informal group structures, of authenticity or power, of organizational structures? This is not to deny the strengths of the method, only to demonstrate that its underlying object models may be insufficient, particulary as the "newer" user system criteria are taken into account.

One can, in a like manner, examine such methods as BSP, ISAC, ETHICS, PORGI, HIPO, Warnier-Orr, Jackson, etc. What one finds is that each has a different set of object system models, reflecting not only the world-view of its developers but also their underlying criteria. This leads to the conclusion that rather than adopting methods for IS development, we should be generating and adopting specific object system contexts as appropriate for the system under development (cf. Welke, 1980).

DEVELOPMENT APPROACH

The term "development approach" is intended to embrace the various aspects and considerations associated with the conduct of (information systems) development. These include, but are not limited to: the definition and sequence of development stages/activities/tasks, the definition of "stakeholders" and their roles in the development process, and the relationship between stakeholders and the development tasks, all umbrellaed under a general IS development mission. The development approach and its mission is, of course, closely coupled with the "IS Development Outlook" discussed above, as well as with the choice of object system contexts. However, as suggested in the introduction, it is helpful to separate the three for purposes of discussion.

First, the term "stakeholder" requires some clarification. A stakeholder (in the development process) is anyone who influences or is influenced by the development process or its results. An actor is a stakeholder who "actively" or "passively" participates in the development process. A passive role is one in which the actor assumes the orientation of a "sink," i.e., if asked a question, he/she will respond with an answer. Interpretation, extrapolation, or speculation is limited. Furthermore, the passive actor does not assume explicit accountability for the correctness or appropriateness of the reply given. An active role is the contrapositive of this.

The "mission" of information systems development also requires clarification. By this we intend the implicit or explicit statement of objectives of development and the "authorized " scope of its domain of change.

Historically, the information systems development approach has been embodied in what is called the "system life cycle." The "stakeholders" were generally viewed as being: the DP-system, the systems analyst, and the user system management. The "active" actors were the systems analysts. The relationships between tasks and actors were simple, authoritative ones. The "mission" of IS development was to discover and formalize DP-able tasks and to transfer them to the DP-system so as to improve the overall effectiveness/efficiency of the organization as measured in cost/benefit terms.

All of this has changed, at least in concept if not in practice. We will explore (again, briefly) some of the changes that have been suggested or put into place.

First, regarding the mission of IS development, both the objectives and domain of change have been broadened. This has already been taken up in the preceding section on information systems development outlooks. Objectives have been and continue to be broadened to subsume a broader value system which admits more stakeholders and more of their values.

Concurrent with this is a broadening of the perceived domain of change, perceived because it has always been there but uncontrolled. This includes concerns for the user work system and the organizational and informal group structures as discussed in the section on the user-system.

A number of alternative approaches to the sequencing of development tasks have been proposed as well as "new" tasks. Regarding task sequencing, the concept of levels of development has been introduced where distinctions are made between the technical system, the system of captive users, the informal group systems, the discretionary users as individuals, and the system of discretionary and captive users. The approaches to these levels can be top-down, bottom-up, middle-out, data-out, and combinations thereof.

The conceptualization of the stages of development has also undergone scrutiny. From the traditional SLC partitioning of tasks, we have added the change-agent development models of Kolb-Frohman and Lewin-Schein as alternatives. Others have taken a problem-solving orientation, e.g., examination, diagnosis, alternative generation and evaluation, selection, and implementation (cf. Welk and Klein, 1980). Still others have suggested a partitioning: generation, the act of producing ideas; validation, establishing criteria and assessing the relevance of what has been generated; and evaluation, assessing the technical and economic implications of the result (cf. Henderson, 1980). The DSS development model of Garrity (cf. Keen and Morton, 1979) suggests yet another set of stages. A more comprehensive sequencing of development activities is given by Kerola (1979) as the PSC Systemeering Model. One can find yet another partitioning and sequencing in the PORGI development approach of BIFOA (cf. Kolb, 1979).

The point to be made is that we are rapidly moving away from a monoular view of systems development stages based upon minor variations to the SLC development paradigm, to a variety of alternative development partitionings and sequencings. Even within the larger notion of development stages, we are introducing alternative approaches. For example, in

the ETHICS method of Mumford and Weir (1979), the concept of dialectic generation is coupled to the examination, diagnosis, etc. sequence above to produce alternatives against which an Hegelian-like synthesis takes place. Garrity's approach to DSS also employs such a device in the generation of alternative normative models.

New stages are also being defined. Lucas and more recently Ginzberg have put forward the notion of "climate analysis" as a predevelopment step. Here the organizational unit is examined with respect to its amenability to change in general and information systems development in particular, in an attempt to preclude implemetation failures with respect to the user system. Another "early" stage hinted at by the literature is the selection of the "mode of development." This has to do with the designation of actors and stakeholders and their relationship to the on-going tasks of development. This leads us to the second issue of who the developers are or should be.

The problem of deciding upon stakeholders and actors and their relationship to development is by no means a simple one. One key variable is the environment of development (as an extension of the concept of climate). In a simple environment, the fundamental stakeholders are stable, with relatively homogeneous expectations; the activities engaged in are recurring in nature with the basic cause-effect relationships understood (or at least agreed upon). By contrast, a complex environment has shifting stakeholders who are relatively heterogeneous with respect to expectations; the tasks are constantly changing, precluding the possibility of clearly identifying cause-effect relationships or standard operating procedures. Clearly, an assessment of the environment should precede the selection of the development approach. However, we are only now beginning to appreciate these differences and to establish heurisitics for dealing with them. Biggs (1979) in his adaptation fo the Vroom-Yetten model suggests one mechanism available to assess this relationship.

Having selected the actors and stakeholders we are still left with "how" to achieve the desired relationships. For the simpler model of developer as creator (Bostrom, 1978), the authoritative role comes easy to most people. But what of the other end of the spectrum -- participation. Here the problems of generation, validation, and evaluation become much more complex. In this regard, a good deal of work has been done by the Europeans. Mumford and Henshall (1979), DeMaio (1980), Bjorn-Andersen (1980), and others have explored this area and offer a number of insights which should be part of every IS developers knowledge.

Yet another aspect of the development approach is the coupling of the development approach with the object system contexts. This is the arena of skills, tools, and techniques. With an object system modelling framework (and hence context) chosen, it is necessary to "fill-in" the model. This is done through the traditional approaches of interviewing, observation, and data collection as well as more neo-traditional techniques such as Decision Assumption Analysis, Interpretive Structural modelling, Nominal Group Techniques, Diary, Role-exchange, Gaming, and a host of other techniques reported in the IS development literature. Furthermore, as our view of the object system context expands and with it our perceived domain of change, new techniques must be added to the "arsenal" to assess cognitive sytle, job satisfaction, envoronment and climate, informal group stractures, etc. One recent attempt to cope with this is found in the PORGI handbook.

EVALUATION

A recurring theme throughout this section (and the paper) has been the notions of criteria, assessment, and evaluation. IS evaluation has not kept pace with advances in either the DP-system, the user system, or the methodologies. This is barely surprising as we are still, as an area, uncovering the myriad of cause-effect relationships associated with an information systems change; the lack of adequate evaluation tools merely reflects the dynamic and emerging nature of the area. In one sense, this is good; it provides us with the flexibility to expand our potential domain of change without having to first adapt the in-place, ingrained approaches to assessment. On the other hand, we also suffer as a discipline and as a profession from this; we have no way to reject useless ideas and concepts except in the field, or alternatively, to identify real contributions except through the political process of large corporation adoption and the academic journal refereeing process.

Still, some progress is being made. New criteria have been proposed and made measurable, empirical work is being undertaken with increasing frequency to uncover a variety of (often counter-intuitive) cause-effect relationships. New variables are being discovered or re-discovered (witness the current interest in cognitive style) and

related to development approaches. Techniques such as NAPSY and approaches based upon fuzzy set theory for the user assessment of qualitative criteria are being tried and used (cf. Land, 1978). An excellent overview of IS evaluation can be found in Kreibel (1979); an anthology on the more traditional approaches to evaluation is contained in Kleijnen (1980); and a comprehensive bibliography on IS evaluation is available from Welk (1978).

IS DEVELOPMENT AS A USER SYSTEM CONTEXT

Somewhat strangely, IS development has not seen itself as a consumer of its own work. Granted, our work can be characterized in terms of a complex environment and thus not as approachable. However, we suffer the same difficulties as the managers with their semi-decomposable tasks; and while we claim to be able to approach these people, we have not really approached ourselves with the same enthusiasm. In-roads have been made however. Computer-based support systems for IS development have been around for some time and continue to grow. We will review, briefly, some of this work.

Perhaps the best known computer-based IS development support tools are the comprehensive automated system development projects. In North America, the most prominent is the ISDOS project. Out of this project has come the computer-based system description language: PSL/PSA. PSL is, in fact, a comprehensive systems description language which asks the user to "see" the object system in terms of its defined objects and relations. Because of the variety of the objects, relations, and properties within PSL/PSA, this is not a real constraint. The only restriction is the hierarchical view of relations between objects as "sub-"objects. The benefits of using it are considerable because of the query and reporting facilities which work against the stored data base description of the system. Working off of this description is a system being developed and tested at the University of Arizona called PLEXSYS, which promises to achieve the original objective of the ISDOS project, namely the automatic generation of a DP-system from a stored statement of requirements.

Other projects with a similar scope include the HEX project in France, the CASCADE project in Sweden, and the PRIDE-ASDM package in the U.S., as well as a number of abandoned projects in various countries.

Less ambitious in scope are the various support tools available for the various stages of development. Automated data-dictionary packages with algorithms for interactively deriving 'third normal form' and other data models, CAD packages for assistance in developing and updating diagrams, interactive screen generators for prototyping and evolutive design, and of course database management systems for flexible support of indeterminate tasks. Specific techniques, such as interpretive structural modelling have computer-packages available to them. Evaluation tools such as NAPSY are computer-based. What is lacking in this area is an attempt to bring these together under one system.

SUMMARY

IS development methodology should be seen as an "eclectic" combination of contexts, development approaches, and criteria, guided by an overall sense of mission and values and directed at changing the relationships between the existing user-and DP-system. It is not enough to merely explore and adopt one or two methodologies any longer; one must be willing to adapt one's current thinking regarding development to embrace the new dimensions which are being constantly added by an ever-broadening sense of the user system and the advances in technology. Methodology can be either proactive in this regard, by adopting new contexts, approaches, criteria, and values; or it can be reactive with these changes forced upon it by pressures from the user and DP enviroment. This choice is that of the developing unit; either way it will occur.

CONCLUSIONS

In this paper we have embraced the triangle of the user system, the DP-system, and the intervening methodology which brings change to both. We have explored not only the individual components and their aspects, but the interactions as well. In doing so we have come to understand the significance of the inter-relationships both in terms of advances in time as well as at a given point in time. We have also discovered that, as a discipline, we understand relatively little about many of these interactions. As such there are large gaps in our knowledge and many areas for fruitful research, especially of a synthetic nature. We have also reached the point of maturation where it is important to ask questions about our underlying philosophy and values.

Ours is a large area and growing larger by the day. As such there is a tendency to opt out of a systemic view and

applaud narrower analytic views which ignore many of the relationships discussed here. If this is the case, do we deserve the title of Information System Development? We would do well to seriously consider the current tendency towards analytics and pseudo-science and away from systemics.

REFERENCES

For a complete set of references, please contact either of the authors at their respective universities.