Systems Development in Information Systems Research

JAY F. NUNAMAKER, JR., MINDER CHEN, and TITUS D. M. PURDIN

MINDER CHEN is Assistant Professor in the Department of Decision Sciences at George Mason University. He received a B.S. in electrical engineering from National Taiwan University, an M.B.A. from National Chiao-Tung University, and a Ph.D. in management information systems from the University of Arizona. Prior to joining George Mason University, he was a lecturer at MIS Department at the University of Arizona (1988–89) and a research associate (1984–88). His research interests include computer-aided software engineering (CASE), expert systems and Hypertext applications, group decision support systems, executive information systems, and organizational modeling. His papers have appeared in *Computer Personnel*, *Data Base*, *IEEE Transactions on Knowledge and Data Engineering*, and *Journal of Management Information Systems*. He has also co-chaired minitracks in CASE-related areas for the Hawaii International Conference on Systems Sciences since 1989.

TITUS PURDIN is Associate Professor at the Department of MIS at the University of Arizona. He holds a B.A. in economics from Ohio State University, a B.S. in computer science from the University of Arkansas, as well as an M.S. and Ph.D. in computer science from the University of Arizona. His research interests include software engineering, systems software, and distributed systems.

ABSTRACT: In this paper, the use of systems development as a methodology in information systems (IS) research is described and defended. A framework to explain the nature of systems development as a research methodology in IS research is proposed. Use of this methodology in the engineering field in general is compared with its use specifically in computer science and computer engineering. An integrated program for conducting IS research that incorporates theory building, systems development, experimentation, and observation is proposed. Progress in several application domains is reviewed to provide a basis upon which to argue that systems development is a valid research methodology. A systems development research process is presented from a methodological perspective. Software engineering, which is the basic method of applying the systems development research methodology, is then discussed. It is the authors' belief that systems development and other research methodologies are complementary and that an integrated multi-dimensional and multimethodological approach will generate fruitful IS research results. The premise is that research contributions can result from systems development, experimentation, observation, and performance testing of the systems under development and that all of these research approaches are needed to investigate different aspects of the research question.

An earlier version of this paper was originally published in the *Proceedings of the Twenty-Third Hawaii International Conference on System Sciences* (IEEE Computer Society Press, 1990).

Journal of Management Information Systems / Winter 1990–91, Vol. 7, No. 3, pp. 89–106.

KEY WORDS AND PHRASES: systems development, research methodology, software engineering.

1. Taxonomy of Research

IN SOME CIRCLES THERE IS A CONCERN ABOUT THE LEGITIMACY of the research aspect of information systems (IS) as an academic discipline. In particular, the value of *system development* as a research methodology has been questioned. The issue of what constitutes valid IS research can be answered by examining what constitutes *research* in general: its *objectives* and its *methods*. In Blake [6], research is defined as a "systematic, intensive study directed toward fuller scientific knowledge of the subject studied." The goal of research in IS is no exception. It is the purpose of this paper to show that an analysis of the objectives of IS research clearly demonstrates the legitimacy and necessity of system development as a research methodology.

Looking at the literature in some detail, it can be seen that the objectives and methods of research are classified in various ways. The following are a few of the research classifications available.

1. *Basic and applied research.* Basic research involves developing and testing theories and hypotheses in response to the intellectual interests of the researcher, rather than for practical reasons. Applied research is the application of knowledge to solve problems of immediate concern [3, 6].

2. *Scientific and engineering research.* There is no logical distinction between the methods used by the engineer and those employed by the pure scientist. Both types of researchers are concerned with confirming their theoretical predictions. However, they differ in the scale of their experiments and their motives. In the engineering approach, the artistry of design and the spirit of "making something work" are also essential [18].

3. *Evaluative and developmental research.* There are two research approaches directed toward solving problems: evaluative and developmental [1]. The developmental type of research "involves the search for (and perhaps construction or synthesis of) instructions" that yield a better course of action [1, p. 24]. Developmental research has largely been ignored by some researchers. However, without research efforts directed toward developing new solutions and systems, there would be little opportunity for evaluative research.

A perspective in some research is that technology is often treated as a variable that is either present or not present. All technology is considered to be equivalent, which it is not! It is often assumed, for example, that all spreadsheets or word processors are equivalent in acceptability to the user.

4. *Research and development.* Development is the systematic use of scientific knowledge directed toward the production of useful materials, devices, systems, or methods, including design and development of prototypes and processes [6]. Hitch and McKean [34] classified development work as exploratory, advanced, engineering, and operational.

5. *Formulative and verificational research.* The goal of formulative research (also

called exploratory research) is to identify problems for more precise investigation, to develop hypotheses, as well as to gain insights and to increase familiarity with the problem area. The goal of verification research is to obtain evidence to support or refute formulated hypotheses [32]. While there is much that is similar in these classifications, each shows its particular bias toward the nature of research. The idea of system development as a research methodology fits comfortably into the category of applied science and belongs to the engineering, developmental, and formulative types of research.

In the following discussion we take the view that research follows a pattern of "problem, hypothesis, analysis, argument." In this view, problems exist in a research domain and are encountered by observation. One forms a hypothesis and then attempts both to confirm and to generalize on the hypothesis through an analysis. This analysis may take forms as varied as formal proofs, developed systems, and opinion surveys. The results of the analysis become the argument (and evidence) in defense of the original hypothesis. Note that this view of *research methodology* permits system development to be a perfectly acceptable piece of evidence (artifact) in support of a "proof," where proof is taken to be any convincing argument in support of a worthwhile hypothesis. System development could be thought of as a "proof-by-demonstration."

In section 2 we argue that the research objectives of IS necessitate a multimethodological approach that integrates theory building, systems development, observation, and experimentation. Examples of the importance of a systems development research methodology to the advancement of several application areas are examined in section 3. Section 4 outlines the makeup and limits of a systems development methodology, and in section 5 this discussion is related to the more general topic of software engineering. A specific example is presented in section 6.

2. A Multimethodological Approach to IS Research

A *RESEARCH DOMAIN* IS THE SUBJECT MATTER UNDER STUDY in a research project. A *research methodology* consists of the combination of the process, methods, and tools that are used in conducting research in a research domain. The framework for research illustrated in Figure 1 is proposed to explain the relationship between research domains and research methodologies. The body of knowledge includes both research domains and research methodologies. A research process involves understanding the research domains, asking meaningful research questions, and applying valid research methodologies to address these questions. Results from a good research project contribute to the body of knowledge by expanding knowledge in a given domain. It is clear that some research domains are sufficiently narrow that they allow the use of only limited methodologies. All algebraic systems, for example, are constrained to use only available axioms, previously derived theorems of that system, and deductive reasoning to extend the system. It is also clear that some research domains are sufficiently broad that they embrace a wide range of methodologies. This is particularly true in engineering and systems where the concept at issue is likely to be viewed

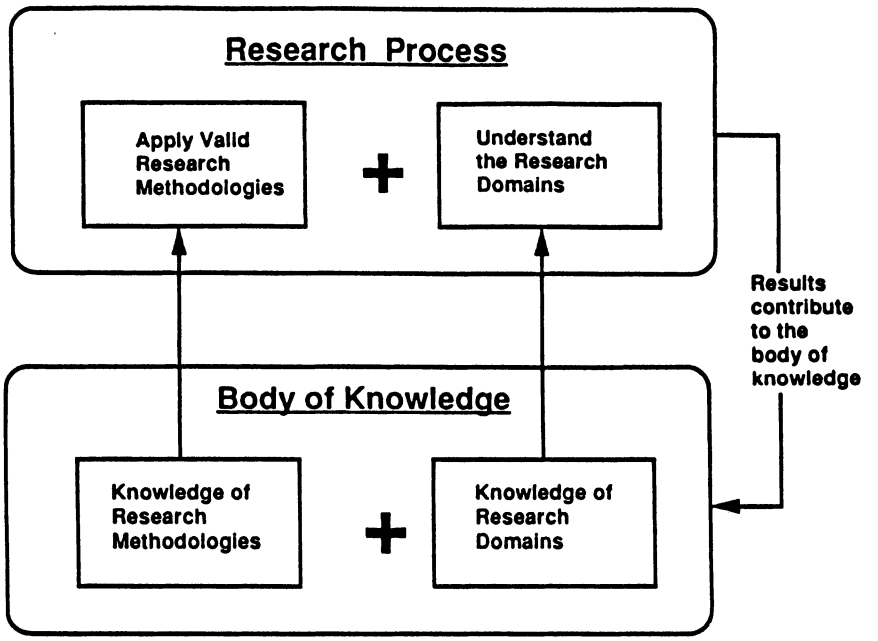

Figure 1. A Framework of Research

for its applications value rather than for its intrinsic value. This suggests that a concept with wide-ranging applicability will go through a research life cycle of the form: concept—development—impact. Much IS research demonstrates such a life cycle. For example, fundamental research in an area such as object-oriented databases eventually contributes to the design and implementation of an experimental database, which in turn leads to research into user acceptance and productivity for the system. A similar case can be made for research efforts in fields such as electrical engineering and computer science.

The pivotal role of system development in this scheme is the result of the fact that the developed system serves both as a proof-of-concept for the fundamental research and provides an artifact that becomes the focus of expanded and continuing research. Contributions at each stage of the life cycle obviously contribute to "fuller scientific knowledge of the subject."

That not everyone has embraced this point of view can be seen in the literature. Benbasat [5] identifies case study, field study, field experiment, laboratory experiment, and sample survey as empirical research strategies for management support systems. In Galliers and Land's [27] taxonomy for IS research methodologies, some newer approaches such as action research were included. However, neither of these analyses included systems development as one of the IS research methodologies. Galliers and Land went so far as to challenge the use of traditional approaches (empirical methodologies) for IS research by stating that although it "may well be

academically acceptable and internally consistent, all too often it leads to inconclusive and inapplicable results" [27, p. 900]. The authors are convinced, however, that without a thorough and complete understanding of a research domain, a researcher may ask the wrong questions or formulate a meaningless hypothesis. No matter what research methods are applied, incorrect or irrelevant questions can only lead researchers to inappropriate conclusions. Where relevant questions and valid hypotheses obtain, systems development may be used as a research methodology.

Having been educated as systems analysts and systems designers, management information systems (MIS) researchers and practitioners understand that the information systems infrastructure of a modern organization permeates all its activities. Only the MIS group within the organization has the necessary systems-building expertise in areas such as database, knowledge base, telecommunications, computer networking, and programming to accomplish systems integration. This natural breadth is reflected in the research domain associated with IS as an academic discipline.

Systems development is an activity found in a number of academic arenas, but unique to MIS within the business school environment. In this larger context, MIS researchers are more properly viewed as systems integrators whose research efforts span a multiplicity of methodologies. These integrated research efforts (often referred to as "projects") can be identified by their relatively long lifetimes and the stages through which they grow (concept—development—impact). Consequently, systems development can be seen not only as a legitimate approach to IS research, but also as a critical contributor among the methodologies available.

The advancement of IS research and practice often comes from new systems concepts. For instance, the use of information systems to support competitive advantages, electronic meetings, executive information systems, concurrent engineering, etc., had its origins in MIS researchers' and practitioners' imagination. This creativity represents research at the "basic" or "concept" level and provides the raw material out of which many large, pragmatic investigations are formed. Concepts alone do not ensure a system's survival. Systems must be developed in order to test and measure the underlying concepts. Systems development is therefore a key element of IS research. Research methodologies, such as laboratory experiments, surveys, and mathematical modeling, are very useful, but not sufficient by themselves to form a well-grounded IS research program.

Perhaps the major motivation in computing and computer application research is, "what can be automated and how can it be done efficiently and effectively?" [2]. This is consistent with the concept—development— impact model. It suggests that "theories" are needed to identify what broad classes of things can be automated, "instantiations" are needed to provide a continuing test bed for the theories, and that "evaluations" of particular instances (systems) are needed to quantify success or failure of a system in both technical and social terms. Systems development provides the exploration and synthesis of available technologies that produces the artifact (system) that is central to this process. The artifact that results from systems development functions as a bridge between the technological research, which we have referred

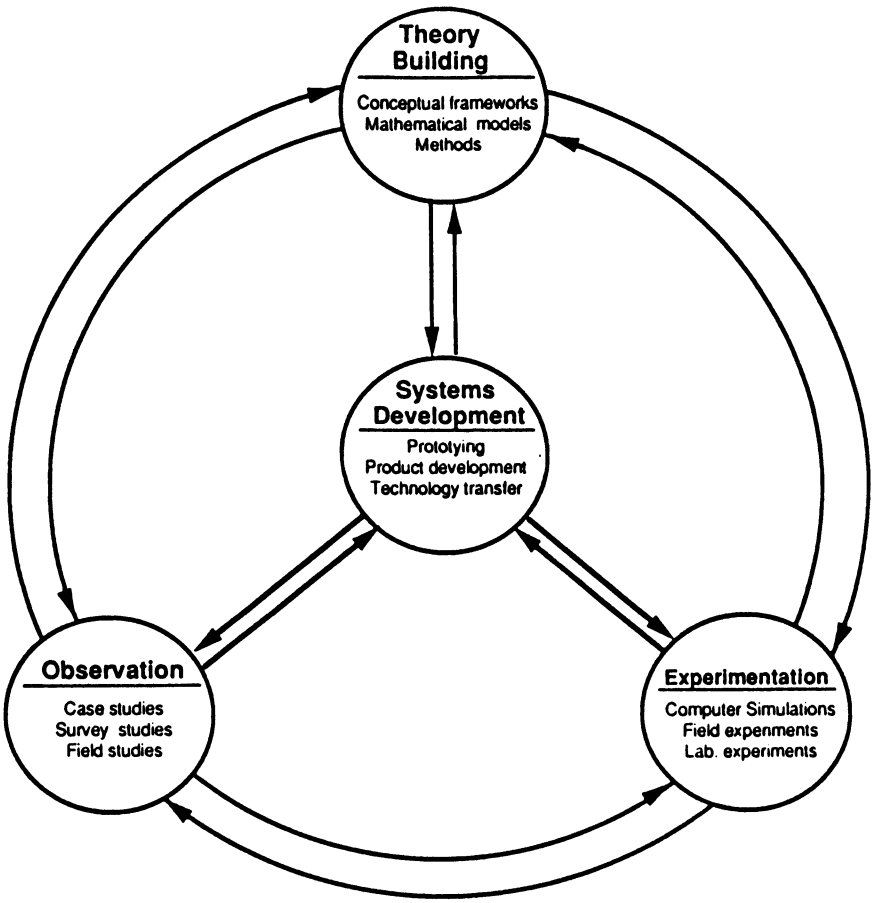

Figure 2. A Multimethodological Approach to Is Research

to as the "concept" stage, and the social research, which we have referred to as the "impact" stage.

The central nature of systems development in the research life cycle is depicted in Figure 2. This shows an integrated approach to IS research, which we believe is necessary if IS research is to keep pace with technological innovation and organizational acceptance. The multimethodological approach to IS research that we propose consists of four research strategies: theory building, experimentation, observation, and systems development. The advantages and disadvantages of these research strategies are discussed below.

1. *Theory building* includes development of new ideas and concepts, and construction of conceptual frameworks, new methods, or models (e.g., mathematical models, simulation models, and data models). Theories (particularly mathematical models) are usually concerned with generic system behaviors and are subjected to rigorous analysis. For instance, mathematical models often have constraining assumptions that

limit the applicability of the models. Because of the emphasis on generality, the outcomes of theory building often display limited practical relevance to the target domain. Relevance refers to potential insights and impacts on practical applications; this suggests that theory building or basic research contributes to the body of knowledge in a research domain but produces nothing (no system) that takes advantage of this new knowledge. Theories may be used to suggest research hypotheses, guide the design of experiments, and conduct systematic observations.

2. *Experimentation* includes research strategies such as laboratory and field experiments, as well as computer and experimental simulations. It straddles the gulf between theory building and observation in that experimentation may concern itself with either the validation of the underlying theories (looking backward along the research life cycle) or with the issues of acceptance and technology transfer (looking forward along the research life cycle). Experimental designs are guided by theories and facilitated by systems development. Results from experimentation may be used to refine theories and improve systems.

3. *Observation* includes research methodologies such as case studies, field studies, and sample surveys that are unobtrusive research operations. Observation is often used when relatively little is known and it is desirable to "get a general feeling for what is involved" in a research domain [59, p. 26]. It may help researchers to formulate specific hypotheses to be tested through experimentation, or to arrive at generalizations that help focus later investigations. Since research settings are more natural, more holistic insights may be gained and research results are more relevant to the domain under study. Researchers are expected to report sufficient contextual and environmental conditions of their research to enable other researchers to judge the limitations of the conclusions.

4. *Systems development* consists of five stages: concept design, constructing the architecture of the system, prototyping, product development, and technology transfer [16]. Concept design is the adaptation and amalgamation of technological and theoretic advances into potentially practical applications. Prototyping is used as a proof-of-concept to demonstrate feasibility. Much systems development research stops at this stage, because it fails to meet initial expectations. Those that are judged successful are expanded into fully articulated production systems. This allows a realistic evaluation of the impacts of the included information technologies and their potential for acceptance. The transfer of technology to organizations represents the ultimate success of those theories, concepts, and systems that complete this race. Difficulties and constraints encountered during the systems development processes can be used to modify the concepts and theories from which the application systems are derived. It is extremely important that other research methodologies be employed to support systems development efforts, because the development of a software system by itself usually is not considered a serious IS research project (in the academic sense).

Systems development is the hub of research that interacts with other research methodologies to form an integrated and dynamic research program. In IS research, no one research methodology should be regarded as the preeminent research paradigm, because no one research methodology is sufficient by itself. In general, where multiple

methodologies are applicable, they appear to be complementary, providing valuable feedback to one another. To gain a complete understanding of a complex research area such as group decision support systems, a multimethodological approach to research is the most effective strategy [66].

3. Systems Development as a Research Methodology

IN THIS SECTION WE SHOW THAT THE PROJECT LIFECYCLE MODEL of research is by no means unique to information systems. It is encountered in many academic disciplines, particularly those in which the research domain includes engineering or systems. The following four examples are intended to highlight the process similarities that exist in divergent research arenas. They were also selected as examples of the interdependence of multiple methodologies.

1. Basic research into the principle of the airfoil and the internal combustion engine allowed the Wright brothers to bring these new technologies together in the first airplane. Aerodynamics and aerostatics, of course, were not recognized research domains at the time. The existence of an airplane, as both proof-of-concept and proof that there were still problems to be solved, served to promote these as important branches of engineering. Continuing experimentation with the technology of flying and the competition of a great variety of completed systems (airplanes) eventually solved many of the early problems, and the technology was eventually transferred to the public marketplace. The aircraft industry is now using the most advanced computer-aided design/computer-aided manufacturing (CAD/CAM) tools to design the next generation of airplanes. These CAD/CAM tools embody theories developed in aerodynamics and heuristics learned from building real systems.

2. In the case of memory management in computer systems, various memory management techniques were developed from previous systems building experiences and evaluation of the systems that were built [19]. First, a virtual memory system was built, then the system was observed in practice. Later, alternative memory management schemes were proposed and simulation was used to study the pattern of memory usage of various memory management schemes. The next step led to the development of mathematical models to study the performance of memory management. Peter Denning, for example, developed the Working-Set Model [20] of program behavior from the observation of the *locality* phenomenon in a paging memory system that was developed by students at the Massachusetts Institute of Technology. Locality is the phenomenon that programs tend to reference main memory in nonuniform and highly localized patterns. "It is an empirical (observed) property rather than theoretical one: [19, p. 222]. A working set memory management policy was proposed to improve systems performance and prevent possible thrashing.

3. The place of electric devices, from radios to computers, as current technologies was well established by the middle of the century. Sophisticated production versions based on vacuum tube technology were in place. The appearance of the transistor as a product of basic research and its application to these existing technologies had a

tremendous impact, both on the nature of the systems and on their acceptance. The transistor, of course, gave way to large-scale integration (circuits on chips), and these same electric devices grow in power and shrink in size at an astonishing rate.

4. In computer-supported cooperative work (CSCW) [31], the advent of electronic mail, teleconferencing [40], and group decision support systems [22, 36] led to research studying the effects of these CSCW tools on organizational structures and dynamics, as well as individual and group behaviors of those who use them. Such empirical studies became possible because production (or very sophisticated research) systems supporting these technologies were available. Electronic mail facilities, for example, were included in the original Internet (ARPA) specification with little regard for how they would be accepted or used. It is easy to argue that today's more sophisticated CSCW systems, such as group decision support systems (GDSS), are products of the serendipitous success of such E-mail facilities.

The first of these examples shows the canonical progression from basic research to "new" systems to established research domains based on those systems. The second demonstrates the feedback of observation on existing systems as problems are identified, generalized, and overcome. In the third example, the impact of continuing basic research and technological advancement can drastically influence existing systems (in addition to or instead of fostering "new" systems). And the last example supports the notion that behavioral research aimed at existing systems can become the impetus for additional, expanded, and improved systems. We maintain that all of these examples showcase the importance of systems development as the focal point of the multimethodological approach to successful research.

4. A Systems Development Research Methodology

METHODOLOGY IS THE PHILOSOPHY OF THE *RESEARCH PROCESS* that "includes the assumptions and values that serve as a rationale for research and the standards or criteria the researcher uses for interpreting data and reaching a conclusion" [3, p. 26]. The research process, the heart of any research methodology, is the application of scientific methods to the complex task of discovering answers (solutions) to questions (problems) [7]. The research process in the social and behavioral sciences can be summarized as follows: (1) choosing the research problem(s), (2) stating hypotheses, (3) formulating the research design, (4) gathering data, (5) analyzing data, and (6) interpreting the results so as to test hypotheses [3, 7]. There are parallels between this social (behavioral) approach to research and the engineering (development) approach described in section 2, although the detailed methods and tools used often differ. Both have much to contribute to the font of information systems knowledge.

The research process outlined in Figure 3 is intended to include elements of both the social and engineering approaches as a token of their compatibility. Research issues that should be addressed in each stage are identified in the figure. The generality of these issues reflects the fact that we believe a systems development methodology is both pivotal and general. In fact, it may well be the case that systems development

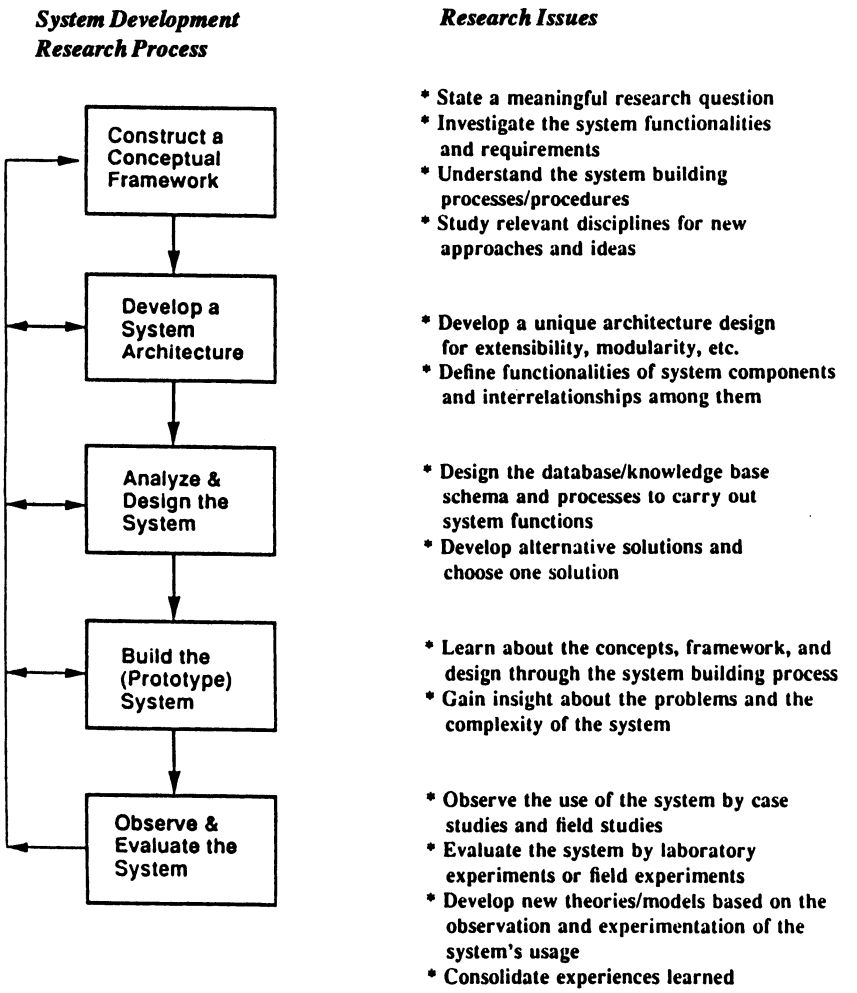

Figure 3. A Process for Systems Development Research

represents a "super-methodology" and actually contains a hierarchy of identifiable "sub-methodologies." For the present discussion, however, the principle parts of a systems development research methodology are addressed below.

1. *Construct a conceptual framework.* Researchers should justify the significance of the research questions pursued. An ideal research problem is one that is new, creative, and important in the field. When the proposed solution of the research problem cannot be proven mathematically and tested empirically, or if it proposes a new way of doing things, researchers may elect to develop a system to demonstrate the validity of the solution, based on the suggested new methods, techniques, or design. This approach is equivalent to a proof-by-demonstration.

Once the system has been built, researchers can study its performance and the phenomena related to its use to gain insights into the research problem. A clear

definition of the research problem provides a focus for the research throughout the development process. The research question should be discussed in the context of an appropriate conceptual framework. Various disciplines should also be explored to find additional approaches and ideas that could be incorporated in the new system.

The conceptual framework leads to *theory building*: different types of theory building efforts in IS research are based on the rigidity of the "theories." (a) Declare the "truth." Something very close to the declaration of a truth is found in Dijkstra's letter to the editor of *Communications of the ACM* in which he declared "go to statement considered harmful" [23]. (b) Formulate a concept (i.e., a framework). Research in this category leads to "a framework that is found useful in organization of ideas and suggesting actions" [62]. Nunamaker and Chen's [54] work on proposing a framework to study software productivity and reusable software components is an example. (c) Construct a method. Parnas's [57] paper on using modularity in systems design basically proposes the concept that it is possible to build a software system with improved flexibility and comprehensibility in a shorter time by using modularization. Some software engineering principles (such as information hiding and hierarchical decomposition) are derived from the concept presented in this paper. Booch's [10] article on "Object-Oriented Development" is also an example of how to construct a new software design method, but at a more specific level. (d) Develop a theory. Halstead [33] has developed a theory, called software science, that calculates the operators and operands of a program to estimate some properties of that program. This is a classic example of building a theory for systems development.

2. *Develop a system architecture.* A system architecture provides a road map for the systems building process. It puts the system components into perspective, specifies the system functionalities, and defines the structural relationships and dynamic interactions among system components. In the development type of research, researchers must identify the constraints imposed by the environment, state the objectives of the development efforts (i.e., the focus of the research), and define the functionalities of the resulting system to achieve the stated objectives. Requirements should be defined so that they are measurable and thus can be validated at the evaluation stage. In the empirical and evaluative type of research, formulating the research hypotheses is an important step in the research process. In the development type of research, researchers usually do not formulate an explicit hypothesis, but they do make assumptions about the research domain and the technical environment for developing the system. Researchers state the system requirements under the constraints of these assumptions, and design and implement the system according to the requirements. Depending on the focus of the research, one might emphasize the new functionalities or innovative user interface features of the proposed new system rather than the throughput or the response time of the system.

3. *Analyze and design the system.* A research project's requirements may be driven by new functionalities envisioned by the researcher, or may be determined partially by the research sponsor's requests. Design, one of the most important parts of a system development process, is rooted in engineering [21]. It involves the understanding of the studied domain, the application of relevant scientific and technical knowledge, the

creation of various alternatives, and the synthesis and evaluation of proposed alternative solutions. A design should be based on theory and abstraction (modeling), which are two other paradigms of computing [21]. Design specifications should be used as a blueprint for the implementation of the system. For a software development project, design of data structures, databases, or knowledge bases should be determined at this phase. The program modules and functions also should be specified at this time, after alternatives have been proposed and explored and final design decisions have been made.

4. *Build the system.* "Building a prototype system is an engineering concept" [62]. Researchers in systems development often conduct their research by building a prototype system. In order to test the system in a real-world setting, however, an effort to further develop a prototype into a product and the transfer of the product into an organization is necessary. Implementation of a system is used to demonstrate the feasibility of the design and the usability of the functionalities of a system development research project. The process of implementing a working system can provide researchers with insights into the advantages and disadvantages of the concepts, the frameworks, and the chosen design alternatives. The accumulated experiences and knowledge will be helpful in redesigning the system. Empirical studies of the functionality and the usability of a system can only be performed after it has been built.

5. *Experiment, observe, and evaluate the system.* Once the system is built, researchers can test its performance and usability as stated in the requirement definition phase, as well as observe its impacts on individuals, groups, or organizations. The test results should be interpreted and evaluated based on the conceptual framework and the requirements of the system defined at the earlier stages. Development is an evolutionary process. Experiences gained from developing the system usually lead to further development of the system, or even the discovery of a new theory to explain newly observed phenomena.

One way of gaining experience is through experimentation. Basili, Selby, and Hutchens [4] provided a framework for conducting experiments in software engineering. A broad overview and many examples of experimental research on human factors in systems development can be found in [15]. Ledgard [42] also discussed some examples of inherent difficulties and the possibility of misleading results in conducting empirical studies of software engineering.

Another method for gaining experience is through observation. Survey studies used in systems development research often focus on the evaluation of different development methods used in a real-world setting. Mahmood's [46] paper comparing the software development life cycle and prototyping methods is an example of using a survey study in the software development domain. Norman and Nunamaker [51] used the survey method to study the CASE productivity perceptions of software engineering professionals. Developing a system is learning by doing. Knowledge gained from the development process can be consolidated into a case study that describes the rationale, process, and experiences learned from developing a system. Orlikowski [56] conducted a case study of the implementation of CASE tools in an organization with an emphasis on their impact

on the IS workplace. IS researchers who conduct a "case study" are usually actively participating in the development or implementation of a system. Such involvement is considered action research [30]. The use of system development as a research methodology in IS should conform to five criteria: (1) the purpose is to study an important phenomenon in areas of information systems through system building, (2) the results make a significant contribution to the domain, (3) the system is testable against all the stated objectives and requirements, (4) the new system can provide better solutions to IS problems than existing systems, and (5) experience and design expertise gained from building the system can be generalized for future use.

5. Software Engineering: A Method for Systems Development Research

IT IS USEFUL TO UNDERSTAND THE CONCEPT OF SOFTWARE PRODUCTIVITY in order to appreciate research efforts in software engineering [9, 54]. Software systems definitely change the way people think and the way they solve problems [44]. For example, the advent of spreadsheet software and financial modeling languages made decision support systems a feasible solution to managerial decision-making problems. Hypertext or similar systems will certainly change the way people read and write, as well as the way they think and communicate [13]. Information systems is an applied discipline. If research in information systems fails to be applicable to real-world problems and opportunities, then IS research efforts are of interest to only a small set of researchers involved in talking to each other [27].

Software can be broadly defined as: (1) the embodiment of the functions of a system, (2) the captured knowledge of an application area, and (3) the information produced during the system development process [26]. Due to the complexity of a software system, its success relies on the application of rigid discipline in its development process, i.e., software engineering.

There is no generally agreed-upon definition of software engineering, but the following definitions will serve as a basis for discussion:

1. Naur's definition [50, p. 9]: "The phrase *software engineering* was deliberately chosen as being provocative, in implying the need for software manufacture to be used on the types of theoretical foundations and practical disciplines that are traditionally in the established branches of engineering."

2. Vick's definition [65, p. ix]: In the preface of *Software Engineering Handbook*, Vick and Ramamoorthy state that software engineering is used to "interpret and apply sound engineering discipline and practice to the design, development, testing, and maintenance of software systems." It is not just "a collection of tools and techniques, it is engineering . . . software engineering can learn from other engineering disciplines."

3. Wegner's definition [67, p. 167]: Wegner emphasizes the conceptual level constructs of software development, saying that "the paradigms of software engineering are those of conventional engineering modified to take into account the fact that software is a conceptual rather than a physical product."

4. Zwass's definition [69, p. 552]: "Software engineering is an emerging discipline of development and maintenance of computer software systems through the creation and use of methodologies and automated tools."

5. Definition in *IEEE Standard Glossary of Software Engineering Terminology* [38, p. 32]: "Systematic approach to the development, operation, maintenance, and retirement of software."

6. Macro and Buxton's definition [45, p. 3]: "The establishment and use of sound engineering principles and good management practice, and the evolution of applicable tools and methods and their use as appropriate, in order to obtain—within known and adequate resource provisions—software that is of high quality in an explicitly defined sense."

7. Humphrey's definition [37]: "Software engineering refers to disciplined applications of engineering, scientific, and mathematical principles and methods to the economical production of quality software." In general, software engineering has the following characteristics: (1) it is an engineering discipline, (2) it studies the methods, techniques, tools, processes, and management of the development of software systems, and (3) it is a systematic approach. Our discussion in this paper is focused on the application of software engineering in IS research and on the study of systems development as a research methodology. Readers who are interested in software engineering education should refer to [24, 28, 29, 58].

The development of (software) systems to conduct research can be traced back to the research paradigm of engineering schools, which has heavily influenced systems development research methodology. Engineers generally agree that "progress is achieved primarily by posing problems and systematically following the design process to construct systems that solve them" [21, p. 10]. The principles of engineering (e.g., [60]) are the foundation of systems development methodology. Engineering discipline also encourages cooperation between theory and practice [58]. Such interactions may help researchers to identify problems and broaden systems development research directions [14]. In addition to building a system in order to demonstrate its feasibility, there are several other goals of software engineering research: (1) measuring properties of systems, (2) improving systems performance, (3) developing formal models of application domains, (4) using specification languages to describe systems behaviors, (5) improving prior systems, and (6) reviewing and synthesizing prior research work [14].

6. Systems Development at Arizona

PARTICIPANTS IN A RECENT COMPUTER SCIENCE AND TECHNOLOGY BOARD (CSTB) workshop urged universities to broaden their view of software engineering research [14]. The authors also encourage IS researchers to recognize the importance of systems development in IS research. At the Department of Management Information Systems (MIS) at the University of Arizona [52], a dominant research theme has been that, because design is the key to IS, emphasis must be placed on rigorous IS

development. Research projects such as PLEXSYS have focused on the building of an integrated environment for systems development [41]. Recognition of the need to allow users, managers, and systems developers to interact in a group setting to elicit IS requirements led to the development of electronic meeting systems [22] at the university. These facilities have been used not only to facilitate the systems development process, but have also been used as settings for a wide range of group meetings (e.g., business planning and knowledge acquisition).

The multimethodological approach has been applied to the department's GDSS research. Conceptual frameworks and theory building efforts have been described in [22] and reports on system development can be found in [11]. Observation was used in a field study that examined the effectiveness of GDSS at IBM [55] and a field study of using GDSS for knowledge acquisition [43]. Several empirical studies have been conducted to validate the effectiveness of the electronic meeting systems [25, 53]. The use of the systems development research methodology in conjunction with other research methodologies in various reference disciplines has been very successful and productive throughout this project. It has provided insight and knowledge to a new area of study (GDSS) in MIS that is changing the way people work. Systems development has been demonstrated to be an important research methodology for conducting IS research. It is the authors' belief that systems development and empirical research methodologies are complementary and that an integrated multi-dimensional and multimethodological approach will generate fruitful research results in IS research. The GDSS facilities at the University of Arizona serve as laboratories for research into the systems of tomorrow.

7. Conclusion

BUILDING A SYSTEM IN AND OF ITSELF DOES NOT CONSTITUTE RESEARCH. The synthesis and expression of new technologies and new concepts in a tangible product, however, can act as both the fulfillment of the contributing basic research and as an impetus to continuing research. The important role played by systems development in the life cycle of complex research demonstrates its credibility as a research methodology. As just one of many available methodologies, systems development takes its place in a multimethodological approach to IS research.

REFERENCES

1. Ackoff, R. L.; Gupta, S. K.; and Minas, J. S. *Scientific Method.* New York: John Wiley & Sons, Inc., 1962.

2. Arden, B. W. (ed.) "What can be automated?" *The Computer Science and Engineering Research Study (COSERS).* Cambridge, MA: MIT Press, 1980.

3. Bailey, K. D. *Methods of Social Research.* New York: The Free Press, 1982.

4. Basili, V. R.; Selby, R. W.; and Hutchens, D. H. Experimentation in software engineering. *IEEE Transactions on Software Engineering* SE-12, 7 (July 1986), 733–743.

5. Benbasat, I. An analysis of research methodologies. In *The Information Systems Research Challenge*, W. F. McFarlan, ed. Cambridge, MA: Harvard Business School Press, 1984, 47–85.

6. Blake, S. P. *Managing for Responsive Research and Development*. San Francisco: W. H. Freeman and Company, 1978.

7. Blalock, A. B., and Blalock, H. M., Jr. *Introduction to Social Research*, second edition. Englewood Cliffs, NJ: Prentice-Hall, 1982.

8. Boehm, B. W. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

9. Boehm, B. W. Improving software productivity. *IEEE Computer*, 20, 9 (September 1987), 43–57.

10. Booch, G. Object-oriented development. *IEEE Transactions on Software Engineering*, SE-12, 2 (February 1986), 211–221.

11. Chen, M. The Integration of Organization and Information Systems Modeling: A Metasystem Approach to the Generation of Group Decision Support Systems and Computer-Aided Software Engineering. Unpublished doctoral dissertation. University of Arizona, Tucson, AZ, 1988.

12. Chen, M.; Nunamaker, Jr., J. F.; and Weber, E. S. Computer-aided software engineering: present status and future directions. *Data Base*, 20, 1 (Spring 1989), 7–13.

13. Conklin, J. A survey of hypertext. *IEEE Computer*, 20, 9 (September 1987), 17–41.

14. CSTB (Computer Science and Technology Board). Scaling up: a research agenda for software engineering. *Communications of the ACM*, 33, 3 (March 1990), 281–293.

15. Curtis, B. (ed.) *Tutorial: Human Factors in Software Development*. Los Alamitos, CA: IEEE Computer Society Press, 1985.

16. Curtis, B. Introduction to empirical research on the design process in MCC's software technology program. In *Empirical Studies of the Design Process: Papers for the Second Workshop on Empirical Studies of Programmers*. MCC Technical Report Number STP-260-87, Austin, TX, 1987.

17. Curtis, B.; Krasner, H.; and Iscoe, N. A field study of the software design process for large systems. *Communications of the ACM*, 31, 11 (November 1988), 1268–1287.

18. Davies, J. T. *The Scientific Approach*, second edition. New York: Academic Press, 1973.

19. Deitel, H. M. *An Introduction to Operating Systems*, second edition. Reading, MA: Addison-Wesley Publishing Co., 1990.

20. Denning, P. J. The working set model for program behavior. *Communications of the ACM*, 11, 5 (May 1968), 323–333.

21. Denning, P. J., et al. Computing as a discipline. *Communications of the ACM*, 32, 3 (January 1989), 9–23.

22. Dennis, A. R.; George, J. F.; Jessup, L. M.; Nunamaker, J. F., Jr.; and Vogel, D. R. Information technology to support electronic meetings. *MIS Quarterly*, 12, 4 (December 1988), 591–618.

23. Dijkstra, E. Go to statements considered harmful. *Communications of the ACM*, 11, 3 (March 1968), 147–148.

24. Dijkstra, E. W. On the cruelty of really teaching computing science. *Communications of the ACM*, 32, 12 (December 1989), 1398–1406.

25. Easton, A. C.; Vogel, D. R.; and Nunamaker, J. F., Jr. Stakeholder identification and assumption surfacing in small group: an experiment study. *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, III (January 1989), 344–352.

26. Freeman, P. *Software Perspectives: The System Is the Message*. Reading, MA: Addison-Wesley Publishing Co., 1987.

27. Galliers, R. D., and Land, F. F. Choosing appropriate information systems research methodologies. *Communications of the ACM*, 30, 11 (November 1987), 900–902.

28. Gibbs, N. E. The SEI education program: the challenge of teaching future software engineers. *Communications of the ACM*, 32, 5 (May 1989), 594–605.

29. Gibbs, N. E., and Fairley, R. E. (eds.) *Software Engineering Education: The Educational Needs of the Software Community*. New York: Springer-Verlag, 1987.

30. Gibson, C. F. A methodology for implementation research. In R. L. Shultz and D. P. Slevin (eds.), *Implementing Operations Research/Management Sciences*. New York: American Elsevier, 1975, 53–73.

31. Greif, I. (ed.) *Computer-Supported Cooperative Work: A Book of Readings*. Palo Alto, CA: Morgan Kaufmann Publishers, Inc., 1988.

32. Grosof, M. S., and Sardy, H. *A Research Primer for the Social and Behavioral Sciences*. New York: Academic Press, 1985.

33. Halstead, M. H. *Elements of Software Science*. Amsterdam: Elsevier North-Holland, 1977.

34. Hitch, C. J., and McKean, R. N. *The Economics of Defense in the Nuclear Age*. Cambridge, MA: Harvard University Press, 1960.

35. Hoffer, J. A. An empirical investigation into individual differences in database models. *Proceedings of the Third International Conference on Information Systems* (1982), 153–167.

36. Huber, G. P. Issues in the design of group decision support systems. *MIS Quarterly*, 8, 3 (September 1984), 195–204.

37. Humphrey, W. S. The software engineering process: definition and scope. *Proceedings of the Fourth International Process Workshop* (May 11–13, 1988).

38. *IEEE Standard Glossary of Software Engineering Terminology*. The Institute of Electrical and Electronics Engineering, Inc., 1983.

39. Ives, B.; Hamilton, S.; and Davis, G. B. A framework for research in computer-based management information systems. *Management Sciences*, 26, 9 (September 1980), 910–934.

40. Johansen, R., and Bullen, C. Thinking ahead: what to expect from teleconferencing. *Harvard Business Review* (March/April 1984), 4–10.

41. Konsynski, B. R., and Nunamaker, J. F., Jr. PLEXSYS: a system development system. In *Advanced Systems Development/Feasibility Techniques*, J. D. Couger, M. A. Colter, and R. W. Knapp, eds. New York: John Wiley and Sons, 1982.

42. Ledgard, H. *Software Engineering Concepts*. Reading, MA: Addison-Wesley Publishing Co., 1987, 111–127.

43. Liou, Y. The Use of a Group Decision Support System Environment for Knowledge Acquisition. Unpublished doctoral dissertation. University of Arizona, Tucson, AZ, 1989.

44. Lyytinen, K. J. Implications of theories of language for information systems. *MIS Quarterly*, 9, 1 (March 1985), 61–74.

45. Macro, A., and Buxton, J. *The Craft of Software Engineering*. Reading, MA: Addison-Wesley Publishing Co., 1987.

46. Mahmood, M. A. System development methods–a comparative investigation. *MIS Quarterly*, 11, 3 (September 1987), 293–311.

47. Mantha, R. W. Data flow and data structure modeling for database requirements determination: a comparative study. *MIS Quarterly*, 11, 4 (December 1987), 531–545.

48. Martin, J., and McClure, C. *Structured Techniques: The Basis for CASE*, revised edition. Englewood Cliffs, NJ: Prentice-Hall, 1988, 4

49. McGrath, J. E. Dilemmatics: the study of research choices and dilemmas. In *Judgment Calls in Research*. J. E. McGrath, J. Martin, and R. A. Kulka, eds. Beverly Hills, CA: Sage Publications, Inc., 1982, 69–102.

50. Naur, P.; Randell, B.; and Buxton, J. N. *Software Engineering: Concepts and Techniques*. Mason/Charter Publishers, Inc., 1976.

51. Norman, R. J., and Nunamaker, J. F., Jr. CASE productivity perceptions of software engineering professionals. *Communications of ACM*, 32, 9 (September 1989), 1102–1108.

52. Nunamaker, J. F., Jr. The MIS research program at the University of Arizona. *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, III (January 1989), 852–862.

53. Nunamaker, Jr., J. F.; Applegate, L. M.; and Konsynski, B. R. Facilitating group creativity with GDSS. *Journal of Management Information Systems*, 3, 4 (Spring 1987), 5–19.

54. Nunamaker, J. F., Jr., and Chen, M. Software productivity: A framework of study and an approach to reusable components. *Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences*, II (January 1989), 959–968.

55. Nunamaker, J. F., Jr.; Vogel, D. R.; Heminger, A.; Martz, W. B.; Grohowski, R.; Mc Goff, C. Experiences at IBM with group support systems: a field study. *Decision Support Systems*. 5, 2 (June 1989), 183–196.

56. Orlikowski, W. J. CASE tools and the IS workplace: findings from empirical research. *Proceedings of the 1988 ACM SIGCPR Conference*, 1988, 88–97.

57. Parnas, D. L. Designing software for ease of extension and contraction. *IEEE Transactions on Software Engineering*, SE-5, 2 (March 1979), 128–138.
58. Parnas, D. L. Education for computing professionals. *IEEE Computer*, January 1990, 17–22.
59. Ray, W., and Ravizza, R. *Methods toward a Science of Behavior and Experience*, second edition. Belmont, CA: Wadsworth, Inc., 1985.
60. Roadstrum, W. H. *Excellence in Engineering*. New York: John Wiley & Sons, 1967.
61. Scacchi, W. Managing software engineering projects: a social analysis. *IEEE Transactions on Software Engineering*, 10, 1 (January 1984), 49–59.
62. Scott Morton, M. S. The state of art of research. In *The Information Systems Research Challenge*, F. W. McFarlan, ed. Cambridge, MA: Harvard Business School Press, 1984, 13–41.
63. Shneiderman, B. *Software Psychology: Human Factors in Computer and Information Systems*. Boston: Little, Brown and Co., 1980.
64. Soloway, E., and Iyengar, S. (eds.) *Empirical Studies of Programmers*. Norwood, NJ: Ablex, 1986.
65. Vick, C. R., and Ramamoorthy, C. V. *Handbook of Software Engineering*. New York: Van Nostrand Reinhold, 1984.
66. Vogel, R. D., and Nunamaker, J. F., Jr. Design and assessment of a group decision support system. In *Intellectual Teamwork: Social and Technological Foundations of Cooperative Work*, J. Galegher, R. E. Kraut, and C. Egido, eds. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., 1990, 511–528.
67. Wegner, P. Paradigms of information engineering. In *The Study of Information*, F. Machlup and U. Mansfield, eds. New York: John Wiley & Sons, 1983, 163–175.
68. Weinberg, G. *The Psychology of Computer Programming*. New York: Van Nostrand Reinhold, 1971.
69. Zwass, V. Software engineering. In *The Handbook of Computers and Computing*, A. H. Seidman and I. Flores, eds. New York: Van Nostrand Reinhold, 1984, 552–567.