Information Systems Frontiers 3:2, 169–183, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems

*Dawn G. Gregg**
School of Management, Arizona State University West,
P.O. Box 37100, Phoenix, AZ 85069-7100 USA
E-mail: Dawn.Gregg@asu.edu

Uday R. Kulkarni and Ajay S. Vinzé
School of Accountancy and Information Management,
College of Business, Arizona State University, P.O. Box 873606,
Tempe, AZ 85287-366, USA
E-mail: Uday.Kulkarni@asu.edu
E-mail: Ajay.Vinze@asu.edu

Abstract. *The Information Systems (IS) discipline, and related research, focuses on the development, understanding, and use of technology to meet business needs. Technology, in particular "software," is the basis for IS research, making software engineering a critical component of research in the IS domain. While the importance of software development is well accepted, what constitutes high quality software engineering research is not well defined. Perhaps this is because some software development clearly is not research and it is hard to distinguish between pure application development, and systems development that pushes the boundaries of knowledge. Sir Karl Popper argued that the scientific quality of research is not based on its empirical method, but on the nature of the questions asked. Our research suggests that software engineering can meet Popper's criteria for scientific research.*

Drawing on well-established research philosophies, we propose a software engineering research methodology (SERM) and discuss the utility of this methodology for contributing to and expanding the IS body of knowledge. We also describe the considerations that need to be addressed by SERM to enhance acceptability of software engineering research in IS. Our suggestions are corroborated with a review of current IS software engineering research reported in leading IS journals.

Key Words. *software engineering, research philosophy, information systems research methodology*

1. Introduction

A major component of the Information Systems (IS) discipline is the engineering of software to meet the evolving needs of organizations and individuals. Software engineering has long been argued to be at the core of IS. IS researchers perform software engineering research to demonstrate the viability of new systems concepts, but existing research paradigms do not fully encompass the issues related to expanding the current capabilities of software systems.

Software engineering as a research methodology differs from the tools, techniques and methods used in the construction of software. Software Engineering Research Methodology (SERM) can be defined as an approach that allows the synthesis and expression of new technologies and new concepts in a tangible product that can the contribute to basic research and serve as an impetus to continuing research (Nunamaker, Chen, and Purdin, 1991). SERM is not research into the software development process. It includes proposing, formalizing and developing software systems to improve the effectiveness and efficiency of processes at the individual and organizational level. SERM creates bridges between human processes and technological capabilities, allowing uses of information that were not otherwise possible.

Although SERM is a significant component of applied IS research, the development of a software system, by itself, is usually not regarded as serious research. There are two likely explanations for this. First, because the philosophical foundations for software

*To whom all correspondence should be addressed.

engineering research in IS are not well defined, there is disagreement as to whether software engineering represents a scientific method of inquiry. Second, no accepted standard and very few rules are available to govern the way software engineering research is conducted. While other research methodologies have clear conventions that define how to conduct experiments or perform case studies, how software engineering research is performed is left almost entirely up to the individual researcher. This creates difficulties for reviewers trying to determine whether a given software development project constitutes high quality research.

Research in IS has traditionally taken either a Positivist/Postpositivist or Interpretive/Constructivist approach (Benbasat, 1989; Cash and Lawrence, 1989; Kraemer, 1991). However, in today's IT based environment, the development and implementation of innovative software in an organizational or individual context does not seem to be adequately covered by these paradigms. While the epistemological and ontological underpinning of these paradigms serve IS research well, the explanations, justifications and methods resulting from these paradigms seem inadequate for SERM. One of the objectives of this research effort is to propose a paradigm that more completely describes research that is performed using SERM approaches.

A second objective of this research is to provide guidelines for conducting IS software engineering research. IS research related to systems development has often borrowed from methods specified in systems engineering literature. However, while construction of innovative systems can be research, systems development by itself is not necessarily research unless a strong theoretical and methodological grounding provides rigor to the effort. To this end, we propose a SERM framework that allows developmental research to be performed using a rigorous methodology. Using this framework, we evaluate current practices and approaches in IS software engineering research efforts and illustrate some best practices for SERM.

In the next section, we discuss software engineering as a research philosophy and compare and contrast it with well-established research paradigms. We propose Sociotechnologist/Developmentalist as a new paradigm that incorporates the assumptions made by the Positivists/Postpositivist and Interpretive/Constructivists and extends them to cover SERM research. In Section 3 presents a more focused framework for SERM that takes a three faceted view to software engineering research and illustrates its utility by mapping it to some of the seminal IS/SERM research efforts. In Section 4, using our framework, we evaluate recently published SERM research in IS journals and extract some best practices these efforts employ. Finally, in Section 5 we summarize our efforts and provide some concluding thoughts related to SERM research.

2. *Software Engineering as Research*

In our attempt at defining software engineering as a research practice, we ask, "Can software engineering be research?" The IS field is founded on the study of information systems and their characteristics and how information systems and programs support human purposes (March and Smith, 1995). The information system and supporting software is fundamentally important in addressing these issues. However, can the engineering of an information system itself constitute a research effort?

It is easy to find extreme cases in which software engineering may not be considered serious research. Obviously, writing a program to solve a trivial problem is not research. Applications of existing software to varied contexts is likewise questionable from a research perspective, even if it supports some ongoing research effort. In these cases, the technology presented does not extend current knowledge boundaries. On the other hand, there are breakthrough software engineering efforts, like the development of databases, for example, that obviously qualify as research. The problem is there is a vast body of software engineering efforts that fall between these two extremes. This brings us back to the initial question: Can software engineering be considered research? And if so, how?

2.1. The nature of software engineering research

Research in its most general form can be described as an approach to one of the many different ways of promoting knowledge enhancement or understanding. Research is inherently different from other ways of knowing such as insight, divine inspiration, and authority dictates (Kerlinger, 1986; Mertens, 1998). Furthermore, scientific research is a process of systemic inquiry conducted under the aegis of a theoretical framework. The use of theory as a basis for inquiry helps distinguish research from other similar efforts that are labeled as *evaluation* (Langenbach, Vaughn, and Aagaard, 1994). The distinction between evaluation and research is important since both can and do make use of systematic inquiry methods. Mertens (1998, p. 3) suggests that while "evaluation is associated with the need for information for decision making in a specific setting, research is more typically associated with generating new knowledge that can be transferred to other settings." The importance of a theoretical framework, although well understood and frequently referred to, is often sidestepped in software development research.

To decide whether software engineering can be considered as research we examine some of the definitions of research. Kerlinger (1986, p. 10) defined (scientific) research as a "systematic, controlled, empirical, and critical investigation of natural phenomena guided by theory and hypotheses about the presumed relations among such phenomena." Mertens (1998, p. 2) defined research as "a systematic inquiry that is designed to collect, analyze, interpret, and use data to understand, describe, predict, or control." Under these definitions, it is difficult to classify the construction of a software system as research.

This, however, is not the only way of viewing research. Sir Karl Popper (1980) argued that the scientific quality of research is not based on its empirical method. He asserts that the value of scientific research is determined by the risk involved in the phenomenon that it is trying to explain. That is, the criterion that should be used to judge the scientific status of research is its falsifiability, or refutability, or testability. Under his definition, research needs to be conducted to answer questions whose answers are not obvious but that can be tested by some means. Blake (1978) defined scientific developmental research as the "use of scientific knowledge directed toward the production of useful material, devices, systems, or methods, including design and development of prototypes and processes." According to these definitions, whether or not something is research depends not only on the method used to test a theory but also on the nature of the questions asked.

Software engineering can perhaps meet Blake's and Popper's criteria for scientific research. Using the linguistic theory context, Lyytinen (1985) argues that a variety of views, ranging from Fregean core (denotational) to Skinnerian response (behavioristic), should be included when studying IS development to better understand the views and goals of the development environment. If a researcher is proposing an entirely new way of looking at a problem and wonders if a system can be developed that will address the problem, then the engineering of such software would constitute research. For example, in the late 1980s researchers were wondering if information systems could be developed to aid group decision making (DeSanctis and Gallupe, 1987). These researchers needed to identify what aspects of group communication and decision making could be improved by the introduction of technology and to show how to configure a system to provide this benefit. Up to that time, it had not been clear that technology could be developed to address the group-decision making problem. Necessary behavioral and system characteristics had not been identified and rules that could be used to control group interaction had not been created. The development of the initial prototype system to support group activities based on "theories" of IS potentially represented a proof-of-concept or proof-by-demonstration.

2.2. Software engineering as a research paradigm

In describing any method as a viable research approach, we need to understand how it stacks up vis-á-vis the major paradigms of research. Paradigms are composed of "assumptions about knowledge and how to acquire it and about the physical and social world" (Hirschheim and Klein, 1989; Hirschheim, Klein, and Lyytinen, 1995). Guba and Lincoln (1994) suggest three questions that need to be addressed in defining a paradigm: What is the nature of reality that is addressed (*ontology*); What is the nature of knowledge (*epistemology*); and What is the best approach to obtaining the desired knowledge and understanding (*methodology*). The two paradigms of interest for information systems researchers, from the field's conception to current times, have been the Positivist/Postpositivist and the Interpretive/Constructivist (Falconer and Mackay, 1999; Probert, 1999; Probert, Rogers, and Moores, 1999).

The Positivist/Postpositivist paradigm follows the empiricist approach, based on the assumption that social worlds are analogous to the natural world, and as such can be studied using similar principles. Reichardt and Rallis (1994) suggest that under this paradigm, explanations of a causal nature can be suggested by grounding these explanations in theory. This paradigm, including such supporting methodologies as experimental, quasi-experimental, correlational and causal comparative (Lather, 1991), are arguably the dominant approach in information systems research today (Morrison and George, 1995).

The Interpretive/Constructivist paradigm suggests that "knowledge is socially constructed by people active in the research process, and that research is a product of the values of the researchers conducting it" (Mertens, 1998; Schwandt, 1994). This paradigm emerged from early efforts in phenomenology and interpretive understanding and is presently also referred to as *hermeneutics* (Eichelberger, 1989). Tesch (1990) identified twenty-six types of methods associated with this paradigm, including enthnographic, hermeneutic, phenomenological and naturalistic. Also included are case studies and other qualitative methodologies that are new to IS research but are lately finding increased acceptance in the IS literature (Markus and Lee, 1999).

While these paradigms provide a good basis for a majority of the IS research stream, they do not fully address the unique requirements of software engineering. A perspective that is common to many IS research projects using the Positivist/Postpositivist paradigm is that (new) technology is a variable that is either present or absent. While studying the impact of technology on organizational processes, groups, and individuals, technology is assumed to be available. The software process that makes it possible for the technology to be applied to the phenomenon under study is assumed away. System development is not considered as part of the fuller understanding of the subject or the building of scientific knowledge. The Interpretive/Constructivist paradigm, on the other hand, prescribes long-term, in-depth observation of the effect of introduction/use of information systems in an organization. Here too, the development of information systems is typically not the focus of such research. Similar to the Positivist/Postpositivist paradigm, the Interpretive/Constructivist paradigm is more concerned in most IS research with the organizational settings and the impact of technology on organizational units. Both these paradigms do not attend to the creation of unique knowledge associated with the development of information systems from their conception to inception.

To alleviate limitations of the Positivist/Postpositivist and Interpretive/Constructivist paradigms and more directly describe the practice of IS software engineering research, we introduce the *Socio-technologist/Developmentalist* paradigm, which addresses the valuable contribution of software systems and associated processes to scientific knowledge building. We then compare and contrast its characteristics with those of the other two paradigms, pointing out that the three approaches to scientific knowledge building represented by the three paradigms are intrinsically interdependent. Table 1 presents the contrasting beliefs associated with the three research paradigms described above.

Within the Socio-technologist/Developmentalist paradigm, reality is technologically created. Socially created multiple realities co-exist and are influenced by the need, acceptance, and/or comfort level of technology. Objectivity in the nature of knowledge is important. Knowledge is coded explicitly and is implicitly experienced through the behavior of the system as interactions take place. The supporting methodology is primarily developmental, starting from idea generation (concept) and design to the system's initial implementation and/or formal description. The methodology's focus is on the technological innovations/extensions

Table 1. *Contrasting beliefs associated with major research paradigms*

Basic beliefs	Positivist/ postpositivist	Interpretive/ constructivist	Socio-technologist/ developmentalist
Ontology • *What is the nature of reality*?	One reality; knowable with probability	Multiple socially constructed realities	Known context with multiple socially and technologically created realities
Epistemology • *What is the nature of knowledge*?	Objectivity is important; researcher manipulates and observes in dispassionate objective manner	Interactive link between researcher and participants; values are made explicit; created findings	Objective/Interactive; Researcher creates the context and incorporates values that are deemed important
Methodology • *What is the approach for obtaining the desired knowledge and understanding*?	Quantitative (primarily); interventionist; decontextualized	Qualitative (primarily); hemeneutical; dialectical; contextual factors are described	Developmental (primarily); focus on technological augmentations to social and individual factors

Source: Adapted from Guba and Lincoln (1994).
Note: The category of Social-technologist/Developmentist is suggested by the authors, and is not part of the original model by Guba and Lincoln (1994).

which are intended to affect individual and organizational experience in a positive manner. The actual outcome of the research under this paradigm may subsequently be studied using methodologies associated with the other paradigms.

In describing the Socio-technologist/Developmentalist paradigm, we find that it is intertwined with the other two paradigms. Nunamaker, Chen, and Purdin (1991) call it the "multi-methodological approach" to IS research. March and Smith (1995, p. 255) suggested that "design science (what we call Socio-technologist/Developmentalist) provides substantive tests of the claims of natural science research."

The Interpretive/Constructivist paradigm creates the research context needed to gain familiarity with a new field, observe subtle relationships, and identify needs. Research conducted within this paradigm results in generating new concepts that feed the other two paradigms. The keyword is "generation." The generation of propositions which may later be tested as hypotheses and the identification of technology needs that may result in construction of useful systems.

The Positivist/Postpositivist paradigm directs dispassionate and objective evaluation of dependencies and relationships. The keyword is "confirmation". The research occurring under this paradigm results in fuller understanding of the subject matter. Through a process of confirmation/falsification of propositions, it lays the groundwork for the step-by-step building of the knowledge base of the field. Certain steps may lead to the need for in-depth, longer-term interaction with the participants/processes, thus feeding into the Interpretive/Constructivist paradigm. On the other hand, such interaction may lead to re-specification of the system whose impact is under review, thus feeding into the Socio-technologist/Developmentalist paradigm.

The Socio-technologist/Developmentalist paradigm allows the creation of new systems and transfer of technology to domains that need them. The keyword is "creation." IS systems can be viewed as social systems that are technically implemented (Hirschheim, Klein and Lyytinen, 1995). The knowledge building takes place through conceptualizing, designing, building of prototypes (as proof-of-concept, proof-by-demonstration) and/or by formal (mathematical, logical) proofs and descriptions. The Socio-technologist/Developmentalist paradigm invariably feeds into the other two research paradigms because the intended (and unintended) impact of the systems need to be scientifically evaluated. The introduction of new technology into the social process sometimes leads to completely new sets of questions, some of which may be answered through experimenting and objective evaluation and others may only be answered through in-depth interactions with the participants.

3. *The SERM Framework for IS Research*

Keen (1980) suggested that it was critical for IS researchers to have a "craftsman's feel for the technology." In particular, Keen was referring to the crafting of information systems to exploit the explosive growth of IS technology in organizations. While software engineering has always been considered important to IS research, researchers are increasingly arguing that software engineering is at the core of the IS discipline. Some researchers take that a step further and consider engineering of software to be a key differentiating factor between IS research and other functional disciplines of business (Vinze et al., 1992). The purpose of this section is to present a framework that describes the facets of quality software engineering research.

3.1. Software engineering orientation to IS research

Before presenting our framework, we focus on how software engineering research efforts fit into the research paradigms presented in Section 2, how the methodology potentially connects the three paradigms, and how it participates in the overall process of knowledge building. In the *generate* portion of the software engineering research cycle, developers interpret and construct realities that can be created using a software system. In this context, researchers, to a large extent, follow the Interpretive/Constructivist paradigm in knowledge creation. Software requirements, needs and descriptions are constructed and merged with the attributes of the setting in which the system will be implemented.

In the *create* portion of the software engineering research cycle, concepts are translated into reality using activities associated with systems development. In this phase, developers rely more heavily on the Socio-technologist/Developmentalist approach, as we have defined it. The creation perspective focuses on building a prototype or formalizing the system and thereby improving IS practices, contributing to knowledge by advancing the state of the art through technical innovation.

Finally, the *confirmation* aspect of software engineering research involves transfer of technology to domains that need it. It involves scientific evaluation and measuring the acceptance of the new information systems. This phase traditionally relies on the Positivist/Postpositivist approach. The thrust here is to rule out alternative explanations for the changes observed once a software system is implemented. As such, there are aspects of *generate, create* and *confirm* in all software engineering efforts. SERM bridges the other two research paradigms by creating new realities for individuals and organizations through the development and introduction of software solutions.

3.2. The SERM framework

Following the Socio-technologist/Developmentalist paradigm, the SERM framework (see Fig. 1) defines three facets of the software engineering research methodology: the conceptual, the formal and the developmental. While development is sometimes equated with software engineering, in the SERM framework, conceptualization or the theoretical grounding of the system requirements is suggested as the focal point of the research effort. The conceptual basis is followed by, either mathematical or logic based formalisms and/or development of a system or system prototype. Both the formal and the developmental approaches are important parts of the methodology and are viewed as different approaches to establishing the proof of the concept.

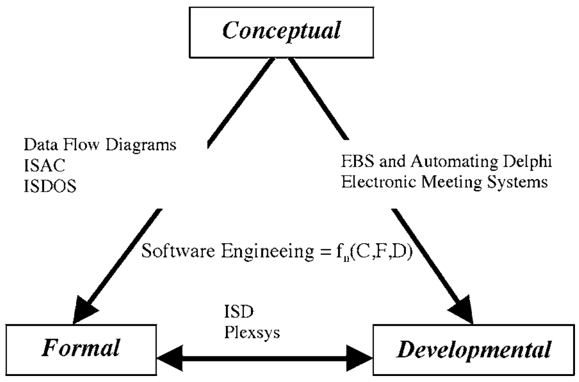

Conceptual
Data Flow Diagrams
ISAC
ISDOS
EBS and Automating Delphi
Electronic Meeting Systems
Software Engineeing = f_n(C,F,D)
ISD
Plexsys
Formal
Developmental

Fig. 1. The SERM framework.

Conceptualization phase. Conceptualization is the fundamental activity in SERM. In this phase the theoretical grounding for the needs and requirements of the research effort are defined. Concepts help researchers think about and communicate ideas via definitions and propositions that are advanced as explanations and predictions for various phenomena and theories. The success of this phase of software engineering depends on (Emory and Cooper, 1991):

1. The clarity with which the researcher details the problem and grounds it with the theoretical constructs, and
2. The understandability and translatability of these concepts.

In describing the conceptual aspect of software engineering research, Nunamaker, Chen, and Purdin (1991) suggest that the "advancement of IS research and practice often comes from new systems concepts" and conceptualization provides the "raw material out of which many large, pragmatic investigations are formed." As an example, they suggest that the use of information systems to support electronic meetings, executive information systems, concurrent engineering as having their origins in IS researchers' and practitioners' imaginations. Under the SERM framework, we suggest that once an idea is conceived and validated the software engineering effort can take on either a formal approach or a developmental approach (or both). Although the sequence of formal and developmental phases is not pre-specified, attempting either of them without the conceptual phase would be inadequate from a research viewpoint.

Formalization phase. Formalization is defined as a mathematical or logic based technique to systematically describe, develop and verify a software system (Bowen and Hinchey, 1995). It is generally agreed that formalisms are an important aspect of software engineering. However, more often than not, they are ignored in the IS literature. The formalization phase of the SERM framework addresses the needs specified in the conceptualization phase, using a mathematical or logic-based explanation. This phase also builds and gives shape to ideas and helps to generalize them. The IEEE has given the following examples of formal methods:

1. A specification written and approved in accordance with established standards,

2. A specification written in a standard notation, for use in proof of correctness.

The use of rules of mathematics and logic is central to the formalizing process as it reduces the possibility of misconceptions and misunderstanding about the system. Formalisms in IS research can also include formal language-based descriptions of the syntax and semantics of the information system envisioned, e.g. Chen's (1976) entity relationship approach to data modeling and Geoffrion's (1987) structured modeling language. It can also include the application of formal techniques such as optimization algorithms and known heuristics to business problems. Other typical activities in this phase include mathematical modeling and evaluation, math/logic proofs, analytical modeling, and computational analysis.

Development phase. When IS research concepts propose a new way of doing things, researchers may elect to develop a system to demonstrate the validity of the solution. The principal approach in this phase is prototyping. By developing a prototype, researchers can study system performance in a controlled environment. Prototyping is an iterative process wherein subsequent developments are based on initial successes. The building of the system is used to demonstrate its feasibility (March and Smith, 1995).

IS researchers often conduct their research by developing a prototype system to demonstrate the feasibility of the design and testing the functionalities of the proposed systems. The process of developing a working system can provide researchers with insights into the advantages and disadvantages of the concepts and the chosen design alternatives (Nunamaker, Chen, and Purdin, 1991). Thus, the accumulation of experiences and knowledge acquired during the systems development process represents a viable research goal in and of itself.

3.3. Illustrating the SERM framework

The SERM framework suggests that software engineering can be considered a conglomeration of three perspectives—conceptual, formal, and developmental. While these phases can be viewed as separate and distinct, we find that for software engineering to qualify as rigorous research, it must address issues in at least two of the three phases.

Numerous IS software engineering research efforts illustrate the combination of traits from two or more of these phases. Some of the work illustrating a linking of the conceptual and the formal phases includes system development methodologies, as in Data Flow diagrams (DeMacro, 1978), ISAC (Lundeberg, Goldkuhl, and Nilsson, 1981) and the ISDOS project (Teichroew and Hershey, 1977). Subsequently, the formal definitional work in ISDOS was extended with a developmental or engineering orientation; software developers translated the first-generation methodologies mentioned above into actual operational systems. As an example, the methodology presented in the ISDOS project was extended to PSL/PSA (Tiechrow and Hershey, 1975) and the Plexsys systems (Nunamaker and Konsynski, 1975; Nunamaker et al., 1976).

A combination of conceptual and developmental aspects of software engineering work is demonstrated in more recent developmental efforts related to group support systems. The GSS development effort draws upon conceptual work on Delphi techniques and stakeholder analysis (Mason and Mitroff, 1973). Instead of following the route of formal definitions, it plunges directly into creating the environments to support group decision activities and scientifically testing their impact (Dennis et al., 1988; DeSanctis and Gallupe, 1987).

Other research projects transition from conceptual formalisms to developmental efforts and can be illustrated with examples from the pioneering research related to databases and data modeling. The development of the relational databases, for example, has as its basis the formalization efforts in relational algebra and normalization theory.

Clearly, several of the foundational efforts in IS can be described as having employed the SERM. However, is current IS research conducted in a manner consistent with the SERM framework and what guidance can the SERM framework offer to IS researchers?

4. Reviewing Current Software Engineering Research Efforts in IS

To examine its utility and to illustrate the various approaches undertaken by IS researchers, current software engineering research efforts in the IS domain were mapped to our SERM framework.

4.1. Data collection and mapping

We focused on software engineering research reported in seven leading IS journals: *Decision Sciences*, *Decision Support Systems*, *Information and Management*,

Table 2. *Research methodologies*

Software engineering
Qualitative
Experimental
Survey
Simulation/modeling
Concept/discussion
Secondary data

Information Systems Research, *Journal of Management Information Systems*, *Management Science*, and *MIS Quarterly*. The time frame for the articles included in this study was from 1996 to 1998. Initially, article abstracts were selected based on a search of the *ABI/Inform* article indexes. The index was searched using keywords such as "software," "program," and "application." The search resulted in identification of 509 articles.

As a first step, the 509 articles were reviewed and categorized according to principle research methodologies according to the seven research methodologies (shown in Table 2) and scanned to see if they considered relevant IS related issues (i.e., the individual or organizational focus of the information system). Articles focusing on purely technical issues (such as data structures, algorithms, hardware considerations, or operating systems) were not evaluated further. The majority of the articles reviewed addressed IS issues out of which 110 were identified as reporting software engineering IS research.

For the 110 software engineering oriented articles, the authors did a preliminary rating of the research to map each one into the SERM framework. This rating involved assessing to what extent the research was conceptual, formal, and/or developmental. Every paper was rated *high*, *medium*, *low*, or *none* on each of the three SERM dimensions. The meaning of these ratings are shown in Table 3.

A rating of *high* in a given dimension indicates that the research meets and/or exceeds the definitions for that research dimension (per Table 3). Only 15 of the 110 articles received a *high* rating on any dimension. A *medium* rating indicates an acceptable performance on that dimension, whereas a *low* rating indicates a minimal consideration given to that dimension.

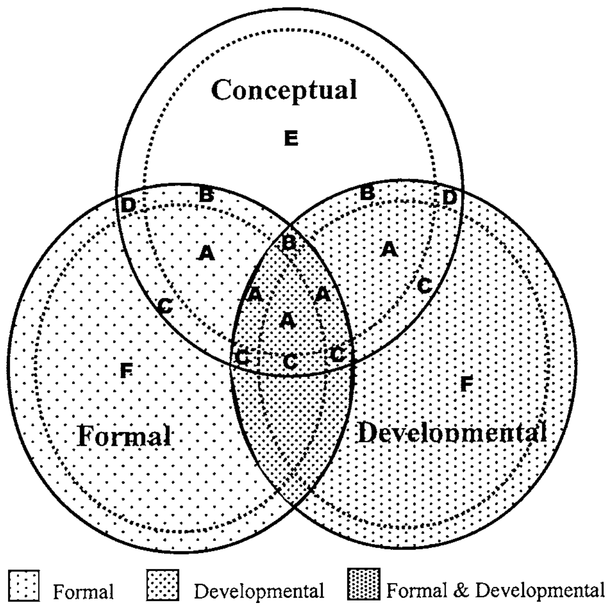
4.2. *Categorizing software engineering research efforts*

Fig. 2 shows a plot of the three overlapping dimensions of the SERM framework. In this plot, articles rated *medium* to *high* for a given dimension fall within the inner (dashed) circle for that dimension; indicating that this research contributed significantly in this dimension. Articles rated *low* for a given dimension are plotted in the narrow region at the edge of the dimension; indicating that this research made a minor contribution in that dimension. Finally, articles rated *none* for a given dimension are plotted outside the outer circle for that dimension. The plot is labeled with letters that are assigned to the software engineering research type associated with each region.

Type A articles represent a high caliber of research in software engineering. They fall within the inner circle of the conceptual dimension and also within the inner circle of at least one of the other two dimensions (formal and/or developmental). The concepts presented in such research increase our knowledge in the given area and significant confirmation is provided by means of either formal methods or developmental prototyping. Type B research represents a substantial extension of

Table 3. *Rating categories for software engineering research*

	Research Dimensions		
Rating	Conceptual	Formal	Developmental
High	Major extensions or generalization of an existing concept or a totally new concept	Defined in math and logic terms; formal definition or proofs; mathematical description.	Prototype or model with validation and verification
Medium	Incremental extension and/or generalization of an existing concept	Definitional without the math and/or logic proofs; establishes correctness criteria.	Prototype or model with limited functionality
Low	Existing concept with limited extensions	Descriptive details and conjectures	Discussion of program requirements
None	No new concept	No formal definitions	No implementation described



A New/Innovative concept with extensive formal and/or developmental research effort
B New/Innovative concept with limited formal and/or developmental research effort
C Limited extension to existing concept with extensive formal and/or developmental research effort
D Limited extension to existing concept with little formal and/or developmental research effort
E New/Innovative concept discussion only
F Development/Formal effort only
No conceptual foundation

Fig. 2. *Types of SERM research.*

current knowledge with some formal and/or developmental effort, while Type C research includes a significant formal and/or developmental effort to demonstrate the viability of a more limited extension of an existing research concept. Type D research represents a limited extension to our current knowledge and is backed up by only limited formal or developmental research effort. Type E and F research was conducted with consideration of only a single dimension of the SERM framework.

The table below shows a cross-tabulation of the conceptual, formal and developmental rankings for the 110 article abstracts deemed IS SERM research.

Table 4. *Formal * developmental * conceptual cross-tabulation*

			Developmental			
Conceptual			None	Low	Medium	High
None	*Formal*	*None*	–	–	4 (F)	–
Low	*Formal*	*None*	7(E)[a]	6(D)	3(C)	3(C)
		Low	2(D)	7(D)	15(C)	1(C)
		Medium	6(C)	1(C)	–	–
		High	1(C)	–	1(C)	–
Medium	*Formal*	*None*	3(E)	2(B	7(A)	2(A)
		Low	5(B)	6(B)	8(A)	1(A)
		Medium	8(A)	3(A)	2(A)	–
		High	5(A)	1(A)	–	–

[a]The letters indicate the type of research per Fig. 2.

Each article count is also labeled A-F, corresponding to the area of Fig. 2 into which the research would fall.

In evaluating the 110 software engineering research papers, 37 were judged to be type A research, the highest quality research according to the SERM framework. These papers presented research that extended the conceptual foundations of the IS field and included a significant formal or development effort. Seventeen of the type A papers were deemed largely formal and represented efforts that extended IS concepts by defining formal rules for specific IS systems, eighteen were deemed largely developmental, i.e. they used a prototyping and testing methodology to demonstrate the appropriateness of the concepts proposed. Finally, two type A papers were both formal and developmental. These papers substantially extended IS concepts, described these concepts using formal methods and demonstrated an IS development effort.

Thirteen papers presented a substantial extension to IS concepts and demonstrated these concepts with limited formal and/or development efforts (type B research). These papers are primarily idea papers and probably represent an initial effort in a stream of research that is to be conducted in a new area. Thirty papers presented limited extensions to IS concepts but demonstrated these concepts with a significant formal and/or development effort (type C research). The research discussed in these papers could represent a onetime effort as opposed to a stream of research.

Twenty-nine papers were judged to be of research type D, E or F. The fifteen type D papers contributed little in the way of new concepts and presented little formal or developmental work. The ten type E papers presented concepts only. In these papers no formal of developmental work was eluded to. The four type F papers presented systems development efforts without presenting their conceptual foundations. These research efforts (type—D, E, and F) could benefit by a grounding the concepts presented more extensively to theory. In addition, the researchers might consider a more extensive formal or developmental extension to demonstrate the viability of their ideas.

4.4. Examples of software engineering researchers in IS

In this section, we present examples of projects that illustrate some of the best practices that are used when conducting SERM research.

Conceptual. Concepts represent the ideas that are critical to the advancement of IS research and practice. Concepts must have strong ties to theory and prior research streams. 53 of the 110 papers reviewed had a strong conceptual foundation.

For example, Sharda and Steiger (1996) proposed using "inductive model analysis" as a means of enhancing a decision-maker's capabilities to develop insights into the business environment. They explored the literature of three distinct disciplines. A strong tie to theory was a critical determinant of this article's overall quality.

In addition to being tied to prior research, an IS concept must also be both innovative and important. Researchable concepts include concepts for:

- Entirely new classes of information systems;
- Significant improvements to existing information systems or classes of information systems;
- Transitioning existing IS concepts into new domains that are significantly different from the original domains.

Examples of articles with a good conceptual foundation are presented in Table 5.

These examples demonstrate that there are many ways to provide a conceptual foundation for IS software engineering research and many reference disciplines upon which this foundation can be based. At a minimum, researchable concepts must have some theoretical or practical basis and should potentially improve IT practice.

Formal. Once a suitable concept is proposed, it is necessary to demonstrate its viability by some means. Researchers can do this using a formal method to systematically specify, develop and/or verify the system concept. Our review of current software engineering research literature indicates that 28 of the 110 software engineering research papers used formal methods in demonstrating their systems' concepts. The formal

Table 5. *Examples of SERM research concepts*

Concept type	Author (year)	Contribution
New classes of information systems[a]	DeSanctis and Gallupe (1987)	Developed early group support systems
	Dennis et al. (1988)	
	Berners-Lee et al. (1994)	Conceived and developed the world wide web and related systems
	Bowman et al. (1995)	
	Borenstein (1996)	Contributed to early electronic commerce practice.
	Brynjolfsson (1997)	
Significant improvement to existing systems or classes of systems	van Harmelen (1997)	Proposed a concept that represented a significant improvement to existing methods for formulating anomalies for knowledge-based systems.
	Batra (1997)	Introduced a mathematically rigorous approach that uses decomposition and synthesis so that relational-concepts can be directly implemented in database management systems.
Transitioning concepts to a new domain	Lenard, Madey, and Alam (1998)	Extended the concept of "hybrid systems" into the auditing domain, combining a statistical model with expert systems.

[a]*Note:* None of the articles for the "new classes of information systems" category were from the 110 software engineering articles reviewed for this study.

Table 6. *Examples of formal SERM research*

Formal method	Author (year)	Contribution
Used established standards	van Harmelen (1997)	Used established standards for conceptual modeling to reinterpret anomalies found in knowledge-based systems and formalized non-rule-based knowledge representation inference structures.
	Gupta and Montazemi (1997)	Developed a distributed connectionist-symbolic architecture using formal system architecture methods.
Used rigorous mathematics and logic techniques including validation	Batra (1997)	Used decomposition and synthesis to allow relational concepts to be used directly in a database management system, that is, without using an intermediate representation like an ER diagram.
	Ong and Lee (1996)	Used abductive logic programming for building a decision support system for the administration of bureaucratic policies.
	Orman (1996)	Developed a new normal form for first order logic to allow all first order constraints to be expressed as counter examples in a database system.
	Xu (1997)	Proposed a method for decision analysis using belief functions in the valuation-based systems that included a decision calculus for belief functions and a formal proof of the benefits.

methods used for any given research can vary, e.g. they can:

- Use established standards for formalizing systems;
- Use mathematical and logical techniques including validation.

Examples of articles using rigorous formal methods are presented in Table 6.

Formal methods are appropriate for a certain class of IS problems. They can be used to extend system capabilities, to improve methods of constructing and evaluating system behavior, or to generalize a concept. When formal methods are suited to the concept being proposed, they represent an appropriate means of conducting software engineering research.

Developmental. Software engineering concepts can also be demonstrated by developing a model or a prototype of a system. Modeling and prototyping are used in two different ways to demonstrate the value of a system. When the system concept represents a substantial change to existing systems, a model or prototype is necessary as a proof-of-concept to demonstrate the feasibility of constructing a new or enhanced system. If the concept hypothesizes that a set of benefits will be provided when an existing technology is used in a new domain, the prototype or model is constructed to demonstrate that the benefits do occur (proof-by-demonstration) (Nunamaker, Chen, and Purdin, 1991). Forty-three of the 110 software engineering research papers performed a significant development effort to demonstrate their systems concepts. Table 7 presents examples of articles to demonstrate the proposed concept.

In the examples in Table 7, generally the development was complete but validation and verification were limited. Further testing of the system in an experimental situation to determine whether the system meets expectations would extend and strengthen such research.

SERM also can include a more extensive validation effort. In the case of Lenard, Madey, and Alam (1998) the value of the research was primarily in the demonstration that hybrid systems represent an improved method for auditors making the going-concern decision. This was accomplished by developing a system prototype and then testing it. They used a three-part experimental methodology that tested auditors' beliefs about expert systems, decision quality with an expert system alone, and decision quality with a hybrid information system.

The development of system prototypes or models is extensively used in software engineering research. If the system concepts are theoretically sound. This represents a valid research methodology. The amount of validation required for these prototypes and models depends on the complexity of the system being developed and on the nature of the research question.

Formal and developmental. Occasionally, SERM articles include both formal and developmental efforts. In these cases the authors usually began with a formal definition of system characteristics and then demonstrated the validity of the formalization by developing a system prototype.

Table 7. *Examples of developmental SERM research*

Development type	Authors (year)	Contribution
Proof-of-Concept	Chen and Sheldon (1997)	Proposed a system architecture that integrates viable design options. It did not include any prototype development, however, the development of the system architecture represented a significant research effort.
	Dean et al. (1997/1998)	Developed prototype electronic-meeting-system modeling tools designed to allow users to work in parallel to contribute directly during meetings.
	Sharda and Steiger (1996)	Proposed a system architecture and built a prototype that integrated several technologies that might help the modeler gain insights from the analysis of multiple model instances.
	Athanasaopoulos (1998)	Built and demonstrated a model that combined data envelopment analysis and goal programming formulations integrated within an interactive planning framework for resource allocation of public services.
Proof-by-Demonstration	Baligh, Burton, and Obel (1996)	Developed and demonstrated the Organization Consultant utilizing useable theory for organizational design.
	Ottoway and Burns (1997)	Presented a conceptual prototype for agent-based structural self-design and described experiments conducted regarding such studies.
	Belz and Mertens (1996)	Developed and tested SIMULEX, a prototype decision-support system for short-term rescheduling in manufacturing.

Ba, Lang, and Whinston (1997) utilized a formal and developmental method in their paper on enterprise decision support using Intranet technology. They presented a knowledge-base enterprise-modeling framework that automatically builds and executes task-specific models in response to user queries. This system represented a new concept based on major modeling approaches in both the DSS and artificial intelligence fields. It included a detailed discussion of knowledge representation issues and the model composition process as well as a partial implementation of an Intranet-based prototype. Another example is an article by Piramuthu, Raghavan, and Shaw (1998). These authors propose a modified learning algorithm for multi-perceptron neural networks to improve the network's classification and prediction accuracy. The efficacy of the proposed method is shown for financial classification.

The use of both formal and developmental methods tends to increase the rigor of software engineering research. When research contains both formal and developmental efforts the demonstration of the underlying concepts is usually better than using either method in isolation.

Implications. SERM research should be multi dimensional involving conceptual innovation coupled with formal and/or developmental efforts. Our review of the literature indicates that the significance and originality of the concept was a critical determinant of the value of the research effort as a whole. The concept provides the tie into theory and the basis for development or formalization.

All the concepts advanced in the "best practice" example articles, highlighted earlier, had ties to theory or to prior research. They extended or generalized an existing concept. These concepts were then "proven" using many different formal or developmental methods, including using established conceptual modeling techniques, formal system architecture methods, decomposition and synthesis, abductive logic, prototype development, model development and system testing. Articles that were judged to have a relatively poor conceptual foundation tended to be presentations of practitioner oriented development efforts as opposed to research conducted using a software engineering methodology.

Researchers engaged in developing and improving IT systems should ensure that: the concepts being advanced have strong ties to theory, the ideas embodied in the system extend current practice and that the contribution of the research is significant for more than solving problems encountered in a single setting.

5. *Conclusions*

In this paper, we investigated research philosophies as a foundation for conducting software engineering research in the IS discipline. The impetus for our effort resulted from the inability to fit software engineering research comfortably into the established research paradigms from the social sciences, namely the Positivist/Postpositivist and the Interpretive/Constructivist paradigms. The assumptions made by these paradigms fail to adequately address the *developmental* orientation of software engineering research. To address this concern, we described a new paradigm, the Sociotechnologist/Developmentalist paradigm, that complements the other two approaches. Using this paradigm, we proposed the SERM framework to address more specifically the issue of standards for conducting software engineering research in IS. In explaining SERM, we cited influential early efforts in IS development as examples. Using the SERM framework, we illustrated the best practices for current software engineering research efforts as reported in some leading IS journals. Our investigation suggests that IS research, for the most part, measures favorably with respect to the SERM framework. However, there are still research projects that tend to equate software development to software engineering research or consist of conceptual discussions without any proof-of-concept. SERM can be used to bring rigor to IS developmental research, and can also be used for critically reviewing current research in IS. Thus it provides a useful metric for software engineering researchers in IS.

References

Athanassopoulos AD. Decision support for target based resource allocation of public services in multiunit and multilevel systems. *Management Science* 1998;44(2):173–187.

Ba S, Lang KR, Whinston AB. Enterprise decision support using Intranet technology. *Decision Support Systems* 1997;20(2):99–134.

Baligh HH, Burton RM, Obel B. Organizational consultant: Creating a useable theory for organizational design. *Management Science* 1996;42(12):1648–1662.

Batra D. A method for easing normalization of user views. *Journal of Management Information Systems: JMIS* 1997;14(1):215–233.

Belz R, Mertens P. Combining knowledge based systems and simulation to solve rescheduling problems. *Decision Support Systems* 1996;17(2):141–157.

Benbasat I (Ed.). *Information Systems Research Challenge (ISRC)—Volume 1: Qualitative Research Methods*. Boston: Harvard Business School, 1989.

Berners-Lee T, Cailliau R, Luotonen A, Neilsen HF, Secret A. The World Wide Web. *Communications of the ACM* 1994;37(8):76–82.

Blake SP. *Managing for Responsive Research and Development*. San Fransisco: W.H. Freeman & Co., 1978.

Borenstein NS. Perils and pitfalls of practical cybercommerce. *Communications of the ACM* 1996;39(6):36–44.

Bowen JP, Hinchey MG. Seven more myths of formal methods. *IEEE Software* 1995;12(4):34–40.

Bowman CM, Danzig PB, Hardy DR, Manber U, Schwartz MF. The Harvest information discovery and access system. *Computer Networks and ISDN Systems* 1995;28(12):119–125.

Brynjolfsson E. A call for exploration: Introduction to special issue on frontier research on information systems and economics. *Management Science* 1997;43(12):1605–1607.

Cash J, Lawrence P. (Eds.). *Information Systems Research Challenge (ISRC)—Volume 2: Experimental Methods*. Boston: Harvard Business School, 1989.

Chen PPS. The entity relationship model—Toward a unified view of data. *ACM Transactions on Database Systems* 1976; 1(1):9–36.

Chen HM, Sheldon PJ. Destination information systems: Design issues and directions. *Journal of Management Information Systems: JMIS* 1997;14(2):151–176.

Dean, DL, Lee JD, Pendergast MO, Hinkey AM, Nunamaker JF. Enabling the effective involvement of multiple users: Methods and tools for collaborative software engineering. *Journal of Management Information Systems: JMIS* 1997/1998;14(3): 179–222.

DeMarco T. *Structured Analysis and Systems Specification*, New York: Yourdon Press, 1978.

Dennis AR, George JF, Jessup LM, Nunamaker JF, Vogel DR. Information technology to support electronic meetings. *MIS Quarterly* 1988;12(4):591–624.

DeSanctis G, Gallupe RB. A foundation for the study of group decision support systems. *Management Science* 1987;33(5):589–609.

Eichelberger RT. *Disciplined Inquiry: Understanding and Doing Educational Research.* New York: Longman, 1989.

Emory CW, Cooper DR. *Business Research Methods,* 4th ed. Homewood IL: Irwin, 1991.

Falconer DJ, Mackay DR. Ontological problems of pluralist research methodologies. In *Proc. 5th AIS Americas Conference on Information Systems*, Milwaukee, WI, August 13–15, 1999: 624–626.

Geoffrion AM. An introduction to structured modeling. *Management Science* 1987;33(5):547–588.

Guba EG, Lincoln YS. Competing paradigms in qualitative research. In: Denzin NK and Lincoln Y.S. eds. *The Handbook of Qualitative Research*. Thousand Oaks, CA: Sage Publications, 1994:105–117.

Gupta KM, Montazemi AR. A connectionist approach for similarity assessment in case based reasoning systems. *Decision Support Systems* 1997;19(4):237–253.

Hirschheim R, Klein HK. Four paradigms of information systems development. *Communications of the ACM* 1989;32(10):1199–1216.

Hirschheim R, Klein HK, Lyytinen K. *Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations*. New York, NY: Cambridge University Press, 1995.

Keen PGW. MIS research: Reference disciplines and cumulative tradition. In *Proceedings of the 1st International Conference on Information Systems*, Philadelphia, Pennsylvania, 1980;9–18.

Kerlinger FN. *Foundations of Behavioral Research*, 3rd ed. New York: Holt, Rinehart & Winston, 1986.

Kraemer KL. *Information Systems Research Challenge (ISRC)—Volume 3: Survey Research Methods*. Boston: Harvard Business School, 1991.

Langenbach M, Vaughn C, Aagaard L. *An Introduction to Educational Research*. Needham Heights, MA: Allyn & Bacon, 1994.

Lather P. Critical Frames in Educational Research: Feminist and Post-structural Perspectives. *Theory into Practice* 1991;31(2):87–99.

Lenard MJ, Madey GR, Alam P. The design and validation of a hybrid information system for the auditor's going-concern decision. *Journal of Management Information Systems: JMIS* 1998;14(4):219–237.

Lundeberg M, Goldkuhl G, Nilsson A. *Information Systems Development—A Systematic Approach*. Englewood Cliffs, NJ: Prentice Hall, 1981.

Lyytinen KJ. Implications of theories of language for information systems. *MIS Quarterly* 1985;9(1):61–74.

March ST, Smith GF. Design and natural science research on information technology. *Decision Support Systems* 1995;15(4):251–266.

Markus ML, Lee AS. (Eds.). Special issue on intensive research in information systems: Using qualitative, interpretive, and case methods to study information technology. *MIS Quarterly* 1999;23(1).

Mason RO, Mitroff II. A program for research on management information systems. *Management Science* 1973;19(5):475–485.

Mertens DM. *Research Methods in Education and Psychology: Integrating Diversity with Quantitative and Qualitative Approaches*. Thousand Oaks, CA: Sage Publications, 1998.

Morrison J, George JF. Exploring the software engineering component of MIS research. *Communications of the ACM* 1995;38(7):80–91.

Nunamaker JF, Chen M, Purdin TDM. Systems development in information systems research. *Journal of Management Information Systems* 1991;7(3):89–106.

Nunamaker JF, Konsynski BR. From problem statement to automatic code generation. In: Lundeberg M. ed. *Systemeering 75*, Studentlilteratur, Lund, Sweden, 1975.

Nunamaker JF, Konsynski BR, Ho TI, Singer C. Computer-aided analysis and design in information systems. *Communications of the ACM* 1976;19(12):674–687.

Ong KL, Lee RM. A decision support system for bureaucratic policy administration: An abductive logic programming approach. *Decision Support Systems* 1996;16(1):21–38.

Orman LV. Constraint by example. *Decision Support Systems* 1996;17(1):3–12.

Ottaway TA, Burns JR. Adaptive agile approaches to organizational architecture utilizing agent technology. *Decision Sciences* 1997;28(3):483–511.

Piramuthu S, Raghavan H, Shaw MJ. Using feature construction to improve the performance of neural networks. *Management Science* 1998;44(3):416–430.

Popper K. Science: Conjectures and refutations, In Klemke ED, Hollinger R, and Kline AD. ed. *Introductory Readings in the Philosophy of Science*. New York: Prometheus Books, 1980:29–34.

Probert SK. Towards a critical framework for IS research. In: *Proc. 5th AIS Americas Conference on Information Systems*, Milwaukee, WI., August 13–15, 1999:172–174.

Probert SK, Rogers A, Moores J. Understanding hard and soft IS development methods: Paradigmatic rigidities or different ends of a spectrum? In: *Proc. 5th AIS Americas Conference on Information Systems*, Milwaukee, WI., August 13–15, 1999: 660–662.

Reichardt CS, Rallis SF. The qualitative/quantitative debate. In: Reichardt CS and Rallis SF. ed. *New Directions for Program Evaluation*. San Francisco, CA: Jossey-Bass, 1994:85–91.

Schwandt TA. Constructivist, interpretivist approaches to human inquiry. In: Denzin NK and Lincoln YS. ed. *Handbook of Qualitative Research*, Thousand Oaks, CA: Sage, 1994:118–137.

Sharda R, Steiger DM. Inductive model analysis systems: Enhancing model analysis in decision support systems. *Information Systems Research: ISR: A Journal of the Institute of Management Sciences* 1996;7(3):328–341.

Teichroew D, Hershey E. PSL/PSA: A computer aided technique for structured documentation and analysis of information processing systems. *IEEE Transactions on Software Engineering* 1977;3(1):41–48.

Tesch R. *Qualitative Research Analysis Types and Software Tools*. New York: Falmer, 1990.

van Harmelen F. Applying rule base anomalies to KADS inference structures. *Decision Support Systems* 1997;21(4):271–280.

Vinze AS, Heltne MM, Chen M, Nunamaker JF, Konsynski BR. Design for Change: Knowledge-Based System Support for Information Centers. *IEEE-Transaction on Systems, Man, and Cybernetics* 1992;22(3):498–512.

Xu H. Valuation based systems for decision analysis using belief functions. *Decision Support Systems* 1997;20(2):165–184.

Dawn G. Gregg is a visiting Assistant Professor at Arizona State University West. She received her Ph.D. in Computer Information Systems from Arizona State University, her M.B.A. from Arizona State University West, and her B.S. in Mechanical Engineering from the University of California at Irvine. Prior to her doctoral studies, she was employed for nine years as a research and development engineer. Her current research focuses on how to organize and maintain Web-based content so that it can be used to better meet business needs. Her work has been published in *Communications of the ACM* and *Decision Support Systems*.

Uday Kulkarni is an Associate Professor of Information Systems in College of Business at Arizona State University. He received his B.Tech. in Electrical Engineering from the Indian Institute of Technology, Bombay, his M.B.A. from the Indian Institute of Management, Calcutta, and his Ph.D. in Management Informations Systems from the University of Wisconsin, Milwaukee. Prior to his doctoral studies, he was worked for five years in corporate planning and control areas. His research includes the use of relational views for decision-support and application of artificial intelligence techniques to manufacturing problems. He has published articles in *IEEE Transactions on Knowledge and Data Engineering, Decision Sciences Journal, Journal of Management Information Systems, Decision Support Systems, and European Journal of*

Operations Research. Professor Kulkarni is a Fellow of the Wakonse Society through his dedication to the teaching profession. He has been recognized for teaching excellence by student groups, his department, and the College of Business at ASU.

Ajay Vinzé is a Professor of Information Systems in the School of Accountancy and Information Management at Arizona State University in Tempe. Prior to joining ASU, he served on the MIS faculty at Texas A&M University. He received his Ph.D. in MIS from the University of Arizona, Tucson in 1988. His research interests include business applications of artificial intelligence technology and the study of computer supported collaborative work. His publications have appeared in leading MIS journals like *Information Systems Research*, *MIS Quarterly*, *Decision Sciences*, *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, *Decision Support Systems*, *Journal of Management Information Systems* and *Omega*. Before joining the academic environment, he was an IT consultant based in the Philippines. He is a member of INFORMS, Association of Information Systems and IEEE Computer society.