MIS Quarterly

SPECIAL ISSUE

A DESIGN THEORY FOR SYSTEMS THAT SUPPORT EMERGENT KNOWLEDGE PROCESSES[1]

By: **M. Lynne Markus**
Management Department
Bentley College
Waltham, MA 02452-4705
U.S.A.
lmarkus@bentley.edu

Ann Majchrzak
School of Business Administration
University of Southern California
Los Angeles, CA 90089-1421
U.S.A.
majchrza@usc.edu

Les Gasser
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
501 East Daniel Street
Champaign, IL 61820
U.S.A.
gasser@uiuc.edu

Abstract

This paper addresses the design problem of providing IT support for emerging knowledge processes (EKPs). EKPs are organizational activity patterns that exhibit three characteristics in combination: an emergent process of deliberations with no best structure or sequence; requirements for knowledge that are complex (both general and situational), distributed across people, and evolving dynamically; and an actor set that is unpredictable in terms of job roles or prior knowledge. Examples of EKPs include basic research, new product development, strategic business planning, and organization design. EKPs differ qualitatively from semi-structured decision making processes; therefore, they have unique requirements that are not all thoroughly supported by familiar classes of systems, such as executive information systems, expert systems, electronic communication systems, organizational memory systems, or repositories. Further, the development literature on familiar classes of systems does not provide adequate guidance on how to build systems that support EKPs. Consequently, EKPs require a new IS design theory, as explicated by Walls et al. (1992).

We created such a theory while designing and deploying a system for the EKP of organization design. The system was demonstrated through

[1]Robert Zmud was the accepting senior editor for this paper.

subsequent empirical analysis to be successful in supporting the process. Abstracting from the experience of building this system, we developed an IS design theory for EKP support systems. This new IS design theory is an important theoretical contribution, because it both provides guidance to developers and sets an agenda for academic research. EKP design theory makes the development process more tractable for developers by restricting the range of effective features (or rules for selecting features) and the range of effective development practices to a more manageable set. EKP design theory also sets an agenda for academic research by articulating theory-based principles that are subject to empirical, as well as practical, validation.

Keywords: IS design theory, IS development, emergent knowledge process, knowledge management

ISRL Categories: AH05, AC04, FA, HA, HD

Introduction

A perennially interesting research topic in the IS field is how to effectively develop new systems. The topic is interesting because, as IT develops and technical knowledge grows, IT is applied to new application areas that were not previously believed amenable to IT support. In the process, new kinds of systems and new development methods are also created.

For example, when IT was first applied to clerical record keeping, the waterfall development method emerged to handle the challenges of building good transaction processing systems (TPS). In the late 1960s, when IT was increasingly applied to managerial reporting and decision making, a new class of systems (decision support systems or DSS) and a new development approach (iterative development) were devised (Keen and Scott Morton 1978). Subsequently, as packaged individual productivity tools became popular, a new development strategy evolved (Cusumano and Selby 1995; Grudin 1991a, 1991b). Similarly, the emergence of executive information systems (EIS) as a distinct class of applications warranted the creation of a new development approach (Watson et al. 1997).

Walls et al. (1992) used the name "IS design theories" to refer to an integrated prescription consisting of a particular class of *user requirements*, a type of *system* solution (with distinctive *features*), and a set of effective *development practices*. Thus, there are design theories for familiar system types, like DSS, TPS, EIS, etc. The benefit of an IS design theory, according to Walls et al., is to articulate the boundaries within which particular design assumptions apply. IS design theories make the design process more tractable for *developers* by focusing their attention and restricting their options, thereby improving development outcomes. In addition, IS design theories inform *researchers* by suggesting testable research hypotheses.

For example, DSS design theory is a contribution to the IS field because it signals system developers to do things differently than they would with TPS. In contrast to the advice given to TPS developers, DSS design theory tells developers *not* to specify decision-making problems as procedural processes and *not* to try to specify all user requirements in advance of starting system development. In so doing, DSS design theory makes the DSS design problem more manageable for developers, and it gives researchers a basis for making predictions about DSS use patterns and impacts. Similarly, software package design theory (Cusumano and Selby 1995; Grudin 1991a, 1991b) helps developers cope with a situation in which they do not have access to users for determining requirements, as they would if they were developing an in-house TPS.

In the spirit of Walls et al., this paper proposes a new IS design theory for a class of user requirements we call emergent knowledge processes (EKPs). Emergent knowledge processes are organizational activity patterns that exhibit three characteristics in combination: "deliberations" with no best structure or sequence; highly unpredictable potential users and work contexts; and

information requirements that include general, specific, *and* tacit knowledge distributed across experts and non-experts. Examples include basic research, new product development, strategic business planning, and organization design. We argue that our design theory is a contribution to the IS literature because EKPs represent an important class of design situations that have not yet been adequately served by existing types of systems and their associated design theories.

The plan of the paper is as follows. In the theoretical background section, we discuss IS design theory and theorizing, describe a theoretically based conceptualization of emergent knowledge processes, and explain why a new IS design theory is needed. Next, we provide background about the development effort that served as the stimulus for our design theory, and we present the design theory as a set of principles that offer guidance to developers. We then explain why EKP design theory represents a contribution to the IS literature, and we discuss its generalizability. Finally, we present an agenda for future research and discuss implications for practitioners.

Theoretical Background

In this section, we first discuss IS design theory and design theorizing. Next, we describe a theoretically based conceptualization of emergent knowledge processes. Finally, we explain why a new IS design theory for EKPs is needed.

Introduction to Design Theorizing

Although "IS design theory" is a term that could refer to general systems theory and the relationship between developers, clients, and users (Churchman 1979) in the abstract, Walls et al. (1992) used the term in a very particular way to refer to solutions for specialized classes of IS design problems, usually given such labels as TPS, DSS, EIS, etc. According to Walls et al., an IS design theory is a package of three interrelated elements: a set of user requirements, a set of system features (or principles for selecting system features), and a set of principles deemed effective for guiding the process of development. By addressing all three elements in conjunction, an IS design theory can be thought of as a complete package of guidance for designers facing particular sets of circumstances.

An IS design theory, as explicated by Walls et al., has two distinctive characteristics: it is based in theory, and it provides guidance to practitioners. The theory underlying an IS design theory (referred to as "kernel theory") may be an academic theory (e.g., organizational psychology) or a practitioner theory-in-use (Sarker and Lee 2002). Kernel theory enables formulation of empirically testable predictions relating the design theory to outcomes like system-requirements fit. For example, it should be possible to establish empirically that the application of DSS design theory to a particular set of requirements produces better results than applying TPS design theory to the same requirements.

At the same time, IS design theories are *normative theories.* That is, they are prescriptive and evaluative, rather than solely descriptive, explanatory, or predictive. Because IS design theories are intended to give guidance to developers, they must not only pass scientific tests of explanatory or predictive power, they must also pass the tests of practice: Does the system work? Does the system do what it was supposed to do? Is the system elegant and aesthetically appealing? (Florman 1994).

It should be emphasized that the IS design theory approach of Walls et al. is *not* a radical departure from established IS practice and theorizing. Its primary contribution is to formalize, justify, and extend the traditional IS practice of labeling system types (e.g., TPS, DSS, GSS, ESS, EIS), describing their characteristic features, and prescribing an effective development approach. The value of an IS design theory is to reduce developers' uncertainty by restricting the range of allowable system features and development activities to a more manageable set, thereby increasing the reliability of development and the likelihood of success, and to stimulate research.

This paper outlines a design theory for systems that support EKP. As mentioned above, a IS design theory in the Walls et al. sense consists of three interrelated elements: (1) a set of user requirements derived from kernel theory, (2) principles governing the development process, and (3) principles governing the design of a system (i.e., specifying and implementing its features). In the next section, we present our kernel theory.

Emergent Knowledge Processes

Work that is to be supported by information technology is generally described in terms of the characteristics of the *process* by which work is performed (Keen and Scott Morton 1978), the characteristics of *users and their work context* (e.g., Markus and Keil 1994), and *users' information requirements* (e.g., related to their critical success factors, cf., Rockart 1984). An IS design theory must address all three characteristics.

The first characteristic, *process,* has traditionally been described in terms of the concept of *structure*. For example, Keen and Scott Morton described three degrees of structure: highly structured, semi-structured, and unstructured. They labeled as "semi-structured" processes like brand management, cash management, and management exception monitoring and distinguished them from "unstructured" processes, such as basic research and the concept definition phase of new product development.

Pava (1983) subsequently pointed out that it is not accurate to say that some processes merely lack structure. After all, as Keen and Scott Morton explained, the degree of structure in processes like cash management can increase over time as more becomes known about them. However, for the managerial decision-making processes of interest to Pava, increased structure is neither possible nor desirable, because it might introduce rigid, stereotyped responses where creativity and flexibility are needed. Such unstructurable processes have been referred to in terms of human *sense-making*: building knowledge in a recursive, participatory, and evolutionary manner (Boland and Tenkasi 1995). Because the term unstructured suggests that structuring is possible and desirable, whereas the term emergent does not, we believe that *emergent* is a better label for many knowledge processes.

An example of an emergent process is new product development, which has been described as a series of trial-and-error experiences in which the developer iterates recursively between problem-finding and solution evaluation (Bhattacharya et al. 1998; Iansiti 1992). Similarly, strategy-making has been described as assemblages of deliberations, with unpredictable triggers and fluid courses, evolving organically as the situation changes (Pava 1983). Finally, the process of organization design has been characterized as one of identifying solutions through analyses of the contingencies, limits, and tradeoffs of alternative organizational patterns and then adapting these patterns when circumstances change (Litterer and Jelinek 1983).

In short, we find the term unstructured insufficient to characterize the processes of new product development, strategy-making, and organization design. Instead, we refer to them as *emergent processes*, in which problem interpretations, deliberations, and actions unfold unpredictably.

The second factor addressed by system designers is *the user*. Most in-house IS development processes (such as for DSS and ES) assume that the user type is known in advance (Grudin 1991a, 1991b; Poltrock and Grudin 1994), permitting systematic requirements analysis. The unpredictability of emergent processes means that it is nearly impossible for a system developer to know in advance the kinds of people who will be called into a deliberation, when they will be called in, or why. In addition, because emergent processes often involve high-level professional and technical personnel, the actors have a high degree of autonomy in how they do their work. They can resist the imposition of standard routines and new technologies (Davenport et al. 1998; Frenkel et al. 1999). Therefore, designers of systems to support emergent processes do not have the luxury of systematic requirements analysis; they must plan

for very infrequent use of support tools (maybe once only) and they cannot even assume that the intended users will want, or can be required, to use their support systems.

An example of "unknown users" arises in strategic planning. For example, Mintzberg (1994) describes strategic planning as "big strategies growing from little ideas" in strange places at unexpected times. Thus, almost anyone in an organization (e.g., line managers, strategic planners, IS specialists) could initiate the process of strategic planning, and almost anyone is a candidate user of a strategic planning support tool. Further, the strategy formulator could work alone or in collaboration with others in the organization. Similarly, in the concept definition phase of new product development, it is often impossible to specify in advance what kinds of experts will be needed to solve a problem. At the same time, it would be counterproductive to limit involvement to the initial members of a new product development team (Clark and Fujimoto 1991). To address these issues, new product development teams routinely pull in experts of many types on an as-needed basis. Finally, in the case of organization design, many different events can trigger the process (Litterer and Jelinek 1983). Some of these triggers are quite rare. As a result, many different kinds of actors *can be* involved—manufacturing engineers, product development engineers, managers, HR specialists, shop floor workers, external consultants—but no one group would *always be* involved, and most people would participate in organization design very infrequently. Further, the different types of potential process participants have considerable autonomy in deciding how to approach the task (such as whether to use experts or even a support tool).

In short, emergent processes are characterized by *highly unpredictable user types* and work contexts. The unpredictability extends to when and why the process is performed and whether support tools will be used.

A third factor considered by system developers is *users' information requirements.* The information requirements of knowledge-intensive emergent processes are quite different from those of semi-structured business processes. First, in many semi-structured business processes, such as brand management, users require systems that analyze numeric data presented in tables and graphs. By contrast, in emergent processes, users must often search for the information they need from documents that are poorly indexed and stored. (See Blair [1984] on the different storage and retrieval capabilities of documents versus data.) Second, as many authorities have noted, much of the knowledge involved in sense-making processes is tacit, not explicit (Weick 1995). As a result, it is difficult to capture and share. Third, knowledge-intensive emergent processes have a high level of expert knowledge content. This means that, when tacit knowledge can be made explicit, it cannot easily be represented numerically, but must instead be represented as if-then rules (Baligh et al. 1996), as cases (El Sawy and Bowles 1997), or as text. And, because non-expert users may not understand expert jargon, the knowledge-base must be translated into terms non-experts can understand (Markus 2001). Finally, in most knowledge-intensive emergent processes, knowledge is distributed across many different people (Hutchins 1991). Some of the distributed knowledge is local (e.g., conditions in a certain geographic locale), and some is general (e.g., scientific knowledge). Unless distributed expertise can effectively be brought together during what Hutchins calls "local design activities," knowledge will be incomplete, and action faulty (Cannon-Bowers et al. 1993).

The challenging knowledge requirements of emergent knowledge processes can be seen in several examples. For instance, Keen and Scott Morton distinguish between the information required for management control and the intelligence required for strategic planning. Strategic planning requires external information, primarily qualitative, with a wide scope and a future time horizon. Aggregate and approximate information is of greater value than detailed and highly accurate data (Mintzberg 1994). In addition, a great deal of intuition and "feel" is involved in managerial decision-making (Mintzberg 1994). Similarly, expert knowledge is a significant factor in new product development

groups (Bhattacharya et al. 1998), and new product development knowledge evolves dynamically (Couger 1996). New product development expertise comprises both codified general knowledge (e.g., scientific rules from physics, mechanics, and chemistry) *and* tacit local knowledge (e.g., concerning such factors as how materials "breathe" in local weather conditions) (Cross 1997). Similarly, the organization design process draws upon general scientific knowledge from disparate disciplines (including organizational behavior, organization design, systems theory, sociotechnical systems design, political behavior, engineering design, and incentive structures) as well as local knowledge, distributed across various actors, concerning union rules, management politics, and the capabilities of individual workers (Litterer and Jelinek 1983).

In sum, then, knowledge-intensive emergent processes have *challenging information requirements*. They require knowledge and expertise in applying the knowledge. They require tacit and explicit knowledge, general and contextual knowledge. Because knowledge is distributed, they require knowledge sharing.

Summarizing across the characteristics of process, user, and information needs, we define an *emergent knowledge process* as an organizational activity pattern characterized by (1) an emergent process of deliberations with no best structure or sequence, (2) an actor set that is unpredictable in terms of job roles or prior knowledge, and (3) knowledge requirements for general and specific distributed expertise. Semi-structured processes may have some of these attributes to a lesser extent, but EKPs are differentiated by having all three characteristics to a significant extent. As we show below, these characteristics make it difficult for EKP support system designers to use IS design theories that were developed for semi-structured decision-making problems.

Why Is a New Design Theory Needed for EKPs?

Many researchers have commented on the poor fit between the requirements of processes with the characteristics of EKPs (e.g., creative problem finding, new product development) and existing IT application types, such as executive information systems (EIS) and expert systems (ES). For example:

- Davenport et al. (1996) argued that, "The abstract and unstructured inputs to and outputs from knowledge work processes...make applications of [information] technology more difficult" (p. 55).

- Stein and Vandenbosch (1996) concluded that, "Advanced IS, in particular ES and EIS, provide ample opportunities for higher-order organizaitonal learning…that is rarely exploited."

- Vandenbosch and Huff (1997) interviewed 36 executive users of EIS and found that only nine of them used their systems for the creative work of scanning (i.e., browsing data to understand trends and relationships and to challenge fundamental assumptions). One problem with EIS was the absence of a predefined model based on expert knowledge of how the data could and should be used.

- Shneiderman (1998) lamented that software tools have had little success in supporting creative problem solving. Although EIS have useful features like information visualization and dynamic queries, they are insufficient as tools for creativity because they rarely allow trying out all permutations of a problem, combining ideas, and rapid prototyping.

- Kivijarvi and Zmud (1993) and Alavi (2000) suggested that ES and DSS requiring codified knowledge will not be successful in domains characterized by subjectivity and complexity. ES require the problem space, the design objectives, and guidelines for addressing the problem space to be defined in advance (Kivijarvi and Zmud 1993), which is difficult to do with EKPs.

- Todd and Benbasat (2000) recently concluded that DSS have successfully supported only the problem solving tasks associated with decision

making. However, problem finding—including the tasks of uncovering the underlying decision problem, gathering relevant information about it, and diagnosing it—is "fuzzy, difficult, and not amenable to technical support" (p. 4).

This poor fit seems to stem from three disconnects between EKP requirements and familiar system types (e.g., DSS, EIS, ES).

The first disconnect concerns the EKP requirement that general expert knowledge (such as about human resource management) must be contextualized when making decisions in particular local conditions (such as extreme conflict or poor morale) (Vandenbosch and Huff 1997). Systems that support semi-structured decision making (DSS and EIS) lack expert knowledge repositories and contextualizing translation rules, thereby inhibiting creative problem finding and solution generation (Todd and Benbasat 2000). Expert systems *do* include general expert knowledge, but they may not support contextualization, and they often sacrifice the flexibility needed for process emergence. The difficulty of reconciling the needs for general knowledge, contextual knowledge, and flexible support leads many authorities to claim that the best way to support EKPs is through personal and electronic communication systems or though document database systems with user-supplied content, such as Lotus Notes (Davenport et al. 1996). But without the discipline provided by validated expert knowledge, organizations as well as communities of practice can degenerate into self-deluding "groupthink" (Brown and Duguid 1998).

A second disconnect is that expert systems, DSS, and EIS are all designed for a known user community (Watson et al. 1997), that is, a known *type* of user. They do not adapt gracefully to the EKP characteristic of shifting user types with widely differing knowledge requirements—a factor that has also been noted in organizational memory systems (Markus 2001). For example, a system that successfully supported in-house technical support consultants was not successfully redeployed among external support providers, and the knowledge-base had to be significantly refined and simplified before it could be successfully used by external customers (El Sawy and Bowles 1997).

Third, knowledge workers are supported today, not with a dearth of tools, but with *too many* tools and with tools that are not integrated. According to a recent Gartner report (Hayward 2000), knowledge workers have access to expert systems, decision support systems, executive information systems, organizational communication systems, organizational knowledge repositories, and collaborative tools. This "tool glut" has two negative results. First, because the tools are not integrated *into the work process*, knowledge workers exhibit constrained and stereotypical work behaviors (Hayward 2000): A fair portion of a knowledge worker's day involves "junk computing" (Guthrie and Gray 1996)—managing tools, instead of getting work done. Second, because the tools are not integrated *with each other*, knowledge workers may not use an essential tool, such as a repository of expert knowledge. For instance, Markus and Keil (1994) found that, when provided with two unintegrated systems—one for product price quotation and another for ensuring high quality product configurations—a computer company's salespeople did not use the configuration tool. Consequently, product configuration quality did not improve.

As a result of these three disconnects, existing IS design theories for DSS, EIS, ES, and groupware do not meet *all three* EKP requirements (process, user, knowledge) simultaneously. Further, while generalized system development methodologies do exist, they do not satisfy the criteria for an *EKP design theory*, because they are not tailored to *EKP* requirements. Developers might be able to apply sociotechnical systems theory (Mumford 1995; Taylor and Felten 1993), participative design (Emery 1993; Greenbaum and Kyng 1991), or contextual design (Beyer and Holtzenblatt 1998) to produce an effective system for a *specific* EKP, but the solution would only be applicable to a particular place, context, and time. Similarly, soft systems methodology was intentionally devised *not* to generate generalizable solutions (Checkland 1984). By contrast, an IS design

theory for EKPs should, according to Walls et al. (1992), define system design and development principles that *generalize to the entire class of EKPs*.

In short, familiar types of systems such as DSS, EIS, ES, and groupware do not individually support all the requirements of EKPs. Applying them to an EKP in combination without integrating them (to the work process and with other tools) is not an adequate solution for reasons of usability and likely effectiveness. Applying a general IS design methodology would solve a particular EKP design problem, but it would not result in a general solution applicable to the class of EKPs. Consequently, a new IS design theory is needed specifically for EKPs.

In such a new EKP design theory, two types of design principles are inextricably intertwined: principles governing the development or selection of *system features* and principles guiding the *development process* (Walls et al. 1992). For example, developing a system that supports executives in responding to new strategic opportunities requires the design principle of *vigilance* (Walls et al. 1992). Vigilance is enabled in a vigilant information system (VIS) by such features as anticipatory alerts, rather than post hoc exception reports. *Designing* a system that embodies vigilance on the part of users requires certain development principles, that is, practices by developers. Rather than, for example, focusing on users' stated information requirements, the developer of a VIS would need to observe executives' response thresholds and triggers. An IS design theory for EKPs represents a similar interweaving of design and development principles related to EKP requirements—an interweaving that is apparent in our presentation of EKP design theory in the next section.

The Top Modeler Case and an EKP Design Theory

In this section, we present our design theory for EKPs in the context of the case that prompted us to develop it. We describe the background of the TOP Modeler system and present the design principles that we evolved during TOP Modeler's development.

TOP Modeler Background

TOP stands for "Technology, Organization, and People" integration. The system called TOP Modeler was developed to support the process of organization design in manufacturing organizations. TOP Modeler was funded with a $3 million grant from the National Center for Manufacturing Sciences and included the active involvement of four companies: Hewlett-Packard, General Motors, Digital Equipment Corporation, and Texas Instruments. (Each company dedicated one person to the project full time for three years.) The second and third authors of this paper managed the system development effort. The first author was only peripherally involved in development (she conducted an assessment of requirements early in the project), providing psychological and emotional distance from the project for reflection and identification of lessons learned.

Organization design is a critical process for manufacturing organizations, since it is known to be associated with good or poor performance on such measures as productivity, cost, quality, and cycle time. Few organizations have been able to infuse organization design expertise throughout their manufacturing operations because of the characteristics of EKPs (process emergence, unpredictable user types and use contexts, and distributed expert knowledge). Few software tools to support the process exist: evaluations indicated that extant tools lacked either a solid base in scientific knowledge about organization design or support for a sociotechnical systems perspective on organization design. For example, the tool developed by Burton presents rules for organizational structure but little specific guidance for redesigning information systems or production and process technologies (Baligh et al. 1996). Consequently, the funding organizations were motivated to underwrite the development of TOP Modeler.

Structurally, the TOP Modeler system has three main components: a knowledge-base of the scientific knowledge about organization design, an inference engine, and an interface for data input and analysis display. The knowledge-base was derived from sociotechnical systems theory (Cherns 1976; 1987; Trist and Murray 1993). Academic experts were commissioned to summarize the empirical literature on specific elements of an organization (e.g., work design, information technology, skills) into sets of rules specifying the appropriate design elements for organizations under a wide variety of conditions. The knowledge content of the system was reviewed and vetted for practical usability through a series of consensus-building meetings held over two years with representatives from the four organizations.

After the knowledge-base was developed, it was evaluated against the eight verification requirements for expert systems: competency, completeness, consistency, correctness, testability, relevance, usability, and reliability (Vermesan 1997). The rules were quantitatively validated using data gathered during intensive three-day site visits to 93 electronics manufacturing companies in the U.S. This massive validation effort (Majchrzak 1997) demonstrated that the knowledge-base was able to predict statistically significant differences between manufacturing firms achieving higher versus lower levels of throughput time, a critical measure of effectiveness for electronics manufacturing firms.

In addition to the knowledge-base validation study, there are several other indicators of system success. First, the system was deployed on time and within budget. Second, the TOP Modeler system was eventually commercialized by TOP Integration, Inc. Third, the system has been used in over two dozen "real use" situations of organization redesign, involving over three dozen naïve users in more than 10 organizations (Majchrzak and Finley 1995; Majchrzak and Gasser 2000).

In a structured evaluation of 19 users (Borys and Majchrzak 1999), the system was deemed helpful in fostering new learning about organization and sociotechnical systems design and in changing the behavior of people involved in organization redesign decisions. In addition, users reported reassessing their business strategies, clarifying business issues, learning more about their organizations, and achieving consensus on important organization design issues. One use of TOP Modeler occurred in a high technology subsidiary that was considering moving its operations from Singapore to Thailand. The system-assisted evaluation revealed a number of weaknesses in the Thai plant, and the planned move was stopped. Another case involved a proposed joint venture between a U.S. manufacturer and a Chinese counterpart. Use of the system helped surface previously unrecognized differences in the strategic directions of these two organizations. The joint venture was postponed until the differences could be resolved.

In short, the evidence suggests that TOP Modeler was successful in supporting organization design. Therefore, the story of TOP Modeler is a worthy exemplar for purposes of design theorizing about emergent knowledge processes.

The Stimulus for an EKP Design Theory for TOP Modeler

According to Walls et al. (1992), one research strategy appropriate for design theory building is action research, coupled with iterative hypothesis development. The action research process starts with requirements derived from kernel theories and hypothesized design and development principles that meet these requirements. The hypothesized principles are then used as the basis for specifying system features. Once developed and deployed, the use and impacts of the system can be observed. If the results are not as expected, new hypothesized principles are generated, a system version instantiating the new principles is developed, deployed, and evaluated, and the cycle is repeated. A desired outcome of action research is better theory.

The TOP Modeler project followed this action research strategy. We started with a kernel

theory. Then, over an 18-month period, the development team repeatedly intervened into the organizational design activities of the involved companies, deploying prototypes that tested various assumptions about how organizational design work is done, observing how users responded, and iterating. Finally, we articulated our learning in the form of a new IS design theory.

At the outset of the TOP Modeler project, we did not use EKP kernel theory, since we had not yet developed the EKP concept. Instead, we initially conceptualized organizational design as a semi-structured decision-making process, following Keen and Scott Morton's (1978) prescriptions.[1] We therefore believed we could effectively use relevant design theories—the theoretical and practical literature on how best to build decision support systems, expert systems, and executive information systems (e.g., Watson et al. 1997). That literature led us to expect that we could identify a target group of users, achieve consensus about their requirements, incorporate experts' knowledge about how the task was best performed and what information was relevant, and design a system that matched the work process. (Table 1 summarizes our initial conceptualization of organization design and the corresponding IS design theory.)

Over time, however, we learned that the IS design theory of semi-structured decision-making processes was inapplicable to the organization design process. The last column of Table 1 summarizes the problems we encountered while attempting to apply such a design theory. (These problems are discussed more fully in the next section.) As a result, we were forced to reconceptualize (1) the *requirements* of the organization design process (as those of an emergent, rather than a semi-structured, knowledge process), (2) the *features* of a system that would adequately support the work of organization design (as a support system that combines general and contextualized knowledge and knowledge sharing capabilities, rather than a DSS, ESS, or EIS), and (3) the *process of developing* such a system (as emergent, rather than iterative). The three related elements of our reconceptualization, summarized in Table 2, underlie a new IS design theory for EKPs.

[1]Decision making requires systematic data searching and subjective analysis. Decision making can be improved by providing IT functionality for data retrieval, reporting, and display. Strategic planning is an example of semi-structured decision making, with the characteristics of low accuracy, future vs. present time horizon, infrequent performance, and need for qualitative and widely scoped information.

EKP Design Theory Principles

This section describes our design theory as a set of six combined design and development principles for EKPs. We present each principle as we evolved it from a design theory for semi-structured decision-making processes to a design theory for EKPs. By describing the evolution of our theory in the context of the TOP Modeler story, our intent is to capture the richness of our design theory in a way that simple verbal guidelines to designers cannot. In addition, exhibit boxes provide details about how the TOP Modeler system and development process exemplify the design principles. Despite this contextual presentation of EKP design theory, our claim is that the theory represents a general strategy for the class of problems we call EKPs.

Principle #1: Design for Customer Engagement by Seeking Out Naïve Users

Early on, we attempted to create a detailed portrait of TOP Modeler's potential users and their information requirements: Which occupational groups would use the system? What did they know, need to know, and not know? How were they likely to use the system? What were the implications for the tool's functionality, interface, and support requirements?

Interviews with representatives from the four sponsoring companies revealed that the potential user set included manufacturing engineers, HR specialists, shop floor workers, and general managers, as well as expert organization development consultants and academics. The interviews

Table 1. Initial IS Design Theory and Problems Encountered While Attempting to Apply it

Initial Requirements *Organization design is a semi-structured expert decision-making process*	**Initial IS Design Theory** *The solution is an expert decision-supporting system, developed via an appropriate iterative development methodology*	**Problems Encountered While Attempting to Apply Initial IS Design Theory**
Users and Their Work Context		
• Organization designers hold identifiable job roles known in advance	• System must support the consensus needs of a known user community, determined through use of a method like RAD	• No identifiable user group; anyone or everyone could perform organization design at any time
Users' Information Requirements		
• Expert organization design knowledge will be useful to lay organization designers	• System must represent experts' knowledge of organization design as if-then rules with prescriptions for action	• Lay organization designers do not use rules and do not follow prescriptions for action
• Expert organization designers see the organization as a unified whole	• Knowledge-base should provide a single, cross-functional view of the organization	• Lay organization designers reinterpret knowledge through their own functional lenses and don't seek to involve other groups or get their input
The Process		
• Expert organization designers follow a prescribed, semi-structured process	• System must incorporate the prescribed process and restrict ad hoc organization designers to the accepted process	• No single process is accepted among experts; lay organization designers can, and do, circumvent prescribed processes

further revealed that we were unlikely to find a common fund of knowledge about organization design among the potential user groups. Further, the range of desired or expected uses of TOP Modeler was quite broad, including:

- Generate design alternatives quickly
- Evaluate designs
- Conduct sensitivity analysis
- Identify impacts of strategic objectives
- Identify organizational and people changes needed for new product and process technologies
- Analyze competitors' capabilities
- Certify suppliers
- Facilitate user involvement in design
- Facilitate learning by a sociotechnical design team about integration of technology, people, and organization factors

Moreover, most of these uses would occur only rarely and unpredictably.

We also learned that there might be resistance to using the system by both potential hands-on users and managers. For example, HR specialists view

Table 2. Revised IS Design Theory for EKPs

Revised Requirements *Organization design is an emergent knowledge process*	**Revised IS Design Theory** *The solution is an EKP support system, developed via an emergent development methodology*
Users and Their Work Context	
• Specific types of users cannot be identified; it cannot be assumed that users will be knowledgeable, trained, or motivated, nor can it be assumed that training and use will be mandated	1. System must be self-deploying; developers should conceptualize each user-system interaction as a customer engagement process and repeatedly seek out "naïve" users through a process of "onion-layering" the design team
Users' Information Requirements	
• Lay organization designers need expert knowledge translated into a form they can use, involving multiple types of tradeoff analyses with clear implications for action • Lay organization designers cannot implement system-recommended actions online; they must convince others to implement organization design changes offline	2. System must translate expert knowledge into actionable knowledge for non-experts; developers should expect to need many functional prototypes, instead of a few nonfunctional prototypes 3. System must induce users to take offline action; developers must observe and strive to change users' offline, as well as online, action
• Lay organization designers must be induced to consider knowledge about other functional areas and to develop a holistic conception of the organization design process	4. System must integrate expert knowledge with local knowledge sharing; multiple needed functionalities must be integrated rather than added
The Process	
• The organization design process is emergent, with many process triggers, many process flows and tradeoff analyses, and many motivations among organization designers • *Many* changes in the process, expert and specific knowledge, and user-system interaction must be expected	5. System must implicitly, not explicitly, guide users' deliberations in desirable directions, without restricting them to a prescribed process; developers should use a dialectical development process instead of a consensus-seeking approach 6. System must be *extremely* flexible; developers should componentize everything, including the knowledge-base

organization design as their unique expertise, and managers expressed concern about obtaining recommendations about organizational design from shop floor workers and supervisors. Finally, the interviews indicated that, despite the complexities of organization design knowledge, users were unlikely to accept any training, either in the organization design process or in system use. The system had to stand totally on its own!

These interviews led us to think of our system as needing to be self-deploying among a population

of reluctant, even hostile, naïve users of every conceivable stripe. An encounter between the system and a naïve user is like a cold sales call on a negatively predisposed prospect: the system had to convert the prospect through a seductive process of *customer engagement*—without the additional support of in-person training, coaching, or consulting.

This principle of self-deployment and customer engagement goes far beyond mere "user-friendliness"—a pervasive design guideline in the DSS literature (e.g., Watson et al. 1997). User-friendliness, which we incorporated into the system from the beginning, did not address people's lack of incentives to use the system (cf., Markus and Keil 1994), nor did it counteract any negative perceptions of the activity the system was designed to support. Therefore, we conceptualized a three-stage customer engagement process to make our system self-deploying: induce naïve users to try the system, provide immediate benefits, and encourage people to stay with the system long enough to complete a sociotechnical analysis.

The first stage required a system that *induced naïve users to try it*. Early on, we elicited knowledge from academic scholars and expert practitioners of organization design and attempted to represent their expert knowledge as the experts represented it. For instance, we initially designed TOP Modeler to *improve organizational effectiveness*, since that objective is the focus of much relevant academic literature. However, through action research with potential non-expert organizational designers (e.g., engineers, shop floor workers), we found that the objective of improving overall organizational effectiveness did not compel them to use a support system. Non-experts would only try a support system if it were focused on the goal of improving specific performance metrics (e.g., throughput times, quality, or new product development ramp-ups) of immediate practical interest to them. We further learned that textual representations of knowledge (e.g., documents describing lessons learned and best practices) did not induce use: these documents were time consuming to read and digest. In short, if we wanted to induce naïve users to try out the system, we needed to model the system on a computer game, with color-coded evaluations of the human side of manufacturing technology, while providing benchmarks to shows about how users' organizations "measured up" to others. How TOP Modeler was designed for this first stage of customer engagement is shown in Exhibit 1.

The second stage of the customer engagement process, we found, required users *to acquire immediate benefits from using the system*. They needed to have their intuitions confirmed quickly or learn something they did not already know. If not, they quickly abandoned the new tool. Exhibit 2 presents how TOP Modeler did this.

The third stage of the customer engagement process involved *encouraging users to stay with the system long enough to complete a thorough sociotechnical analysis*. In TOP Modeler, users were encouraged to stay by initializing all system values to "no"; that is, the default organization was shown to contain none of the required organization features. Only by working through a thorough analysis could many gaps be removed, allowing the user to focus on those that must be fixed.

The three-stage customer engagement model required a major shift in the development process. Initially, the development team followed a traditional user-centered methodology. The project had an Executive Steering Committee (high-level managers from NCMS and each of the four companies and the second and third authors as project leaders), a Development Team (five computer scientists), and a Domain Team (full-time user representatives from each company). The Domain Team's original responsibility was to represent potential users and review prototypes in an iterative development methodology, as suggested by DSS design textbooks (e.g., Watson et al. 1997), using user-centered techniques such as joint design meetings and cooperative prototyping (Greenbaum and Kyng 1991).

We learned that this development process was not compatible with the design principle of self-

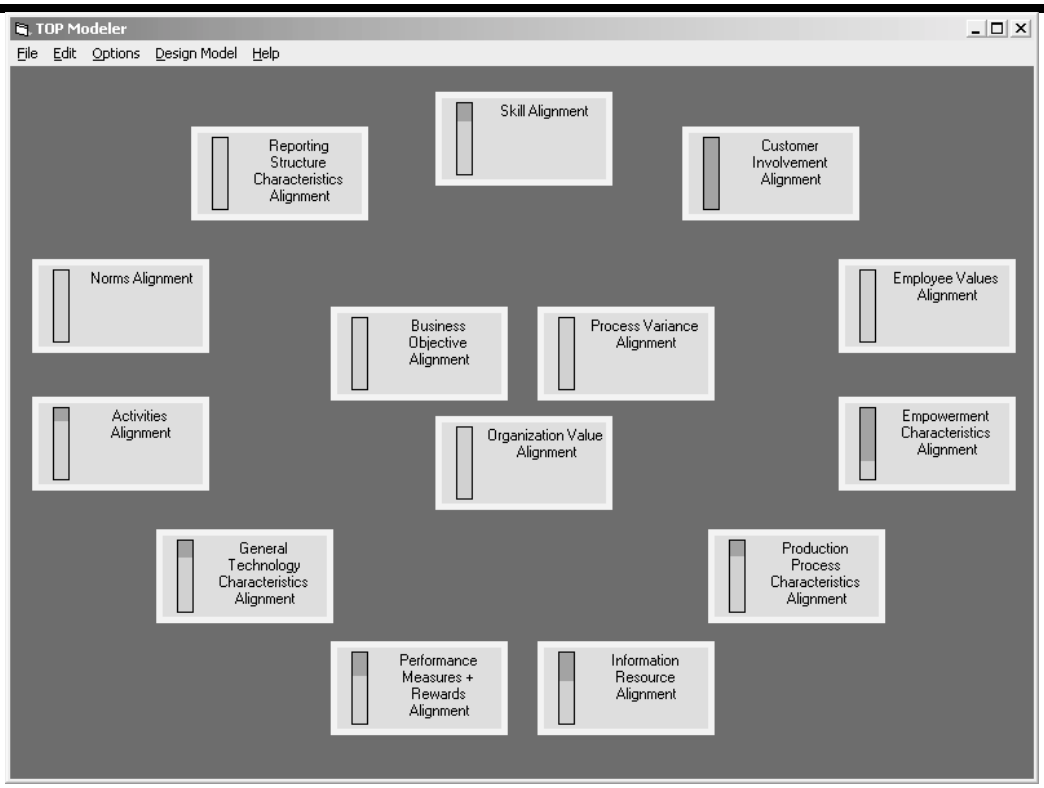
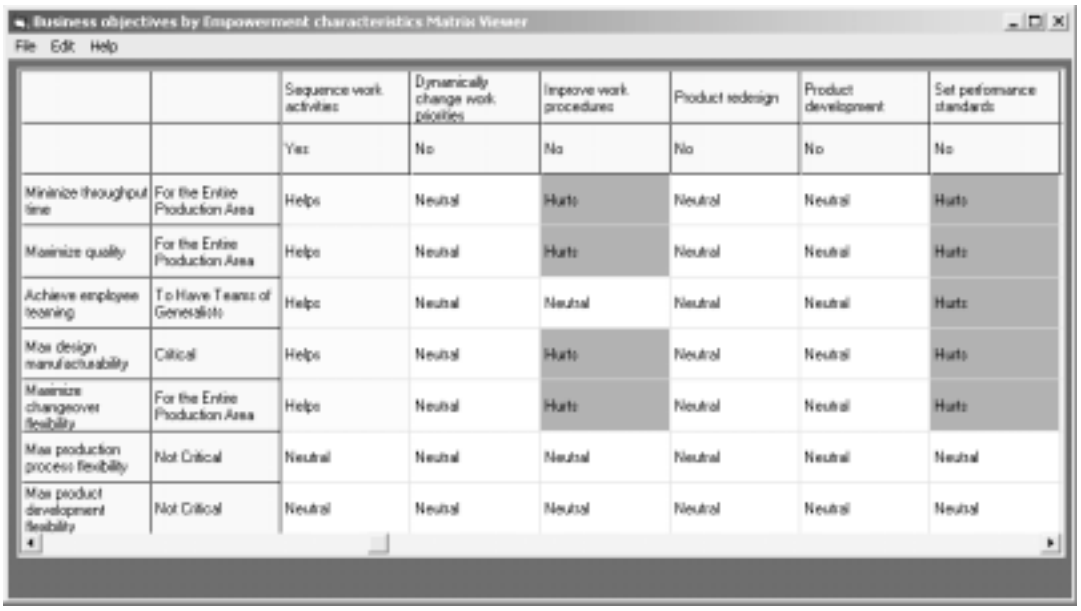


Figure 1. The Ferris Wheel

Exhibit 1: How TOP Modeler Induces Naïve Users To Try It

Potential gaps in users' knowledge were signaled by using color-coding to portray an undesirable initial state. For example, TOP Modeler's primary interface is the Ferris Wheel, shown in Figure 1. The Ferris Wheel presents 12 sets of organization design features laid out around the outside of a circle, where the inside is composed of business strategies (21 process variance control strategies and seven manufacturing organization objectives). The 12 feature sets include norms, skills, customer involvement, discretion, organizational values, employee values, production process characteristics, reporting structure characteristics, technical system characteristics, performance measurement and rewards, information resources, and production tasks. (Each feature set in turn contains from five to 30 separate specific features. For example, a "Skill" feature was "Reading and Writing Capabilities".)

The layout of the Ferris Wheel was intended to convey to naïve users the entire array of appropriate considerations in successful organization design, which we hoped would stimulate their desire to learn about concepts unfamiliar to them. Next to each feature set on the Ferris Wheel were color-coded "thermometers," indicating the percentage of features in the set matched to benchmarked best practices given the business strategies designated by the user. Thus, TOP Modeler was designed to capture naïve users' attention quickly.



		Sequence work activities	Dynamically change work priorities	Improve work procedures	Product redesign	Product development	Set performance standards
		Yes	No	No	No	No	No
Minimize throughput time	For the Entire Production Area	Helps	Neutral	Hurts	Neutral	Neutral	Hurts
Maximize quality	For the Entire Production Area	Helps	Neutral	Hurts	Neutral	Neutral	Hurts
Achieve employee teaming	To Have Teams of Generalists	Helps	Neutral	Neutral	Neutral	Neutral	Hurts
Max design manufacturability	Critical	Helps	Neutral	Hurts	Neutral	Neutral	Hurts
Maximize changeover flexibility	For the Entire Production Area	Helps	Neutral	Hurts	Neutral	Neutral	Hurts
Max production process flexibility	Not Critical	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral
Max product development flexibility	Not Critical	Neutral	Neutral	Neutral	Neutral	Neutral	Neutral

Figure 2. Example Tradeoff Matrix

Exhibit 2. How TOP Modeler Provides Immediate Benefits

Tradeoff matrices provided immediate feedback. The tradeoff matrices were organized around 12 sets of organizational features, with the specific features of each set as column headings and the business strategies as row headings. An example tradeoff matrix is shown in Figure 2. As users inputted a value for an organizational feature, the system evaluated that input for alignment with the organization's business strategy, shown in green. If not aligned, the cell immediately turned red. This feedback encouraged users to continue using the system.

deploying customer engagement. The Domain Team soon became so knowledgeable about the system (because we were involving them in daily conversations and weekly prototyping) that they lost their representativeness as "naïve users"; they had effectively "crossed cultures" and become developers. Therefore, the development process continually needed *new* potential users to interact with the evolving system. Otherwise, we could not accurately assess the system's potential to attract users who had not been previously exposed to it. Exhibit 3 describes how TOP Modeler's development process continually sought out naïve users.

Principle #2: Design for Knowledge Translation Through Radical Iteration with Functional Prototypes

The literature on expert system development (e.g., Durkin 1997; Nikolopoulos 1997) recommends matching the structure of a knowledge-base to the knowledge representation of domain experts. Much organization design knowledge is represented in the scientific literature as if-then heuristics for predicting organizational success. For example, Galbraith's (1977) work on information processing yields rules like: "If an organization experiences high input uncertainty, then jobs should be designed with a high degree of discre-

Exhibit 3. How the Development Process Continually Sought Naïve Users

The Domain Team's responsibilities were shifted from representing users to managing a process that was called "onion layering." As the naivety of Domain Team members faded (and they became less like new customers and more like partners and developers), they were asked to add a new layer of users who would test-drive prototypes and provide feedback. As that second layer of naïve users became too enculturated into the development process to maintain their naïveté, the Domain Team was asked to add yet another new layer of naïve users. By the end of the project, *four* layers of "naive users" had been involved in the project. By never actually replacing the previous users, we were able to maintain their commitment to the project while simultaneously enlarging our user community. Our development process then required not only a user group, but also a continuously replenished set of naive users.

Exhibit 4. How Top Modeler Translates Expert Knowledge into Tradeoffs

The tradeoff matrices were developed from the 1,500 if-then rules of organization design experts. Each cell in each matrix is essentially three rules: one rule for each of the four possible outcomes in each cell (hurts, helps, critical, or neutral). The matrices are designed so the user can trade off different outcomes: "If I don't provide this skill, how much will it hurt me?" versus "If I provide this skill, how much will it help me?" For example, the matrix shown in Figure 2 indicates that a particular business manager is providing inadequate discretion to workers, so that workers' output is being hurt. The manager can then evaluate various options: provide the degree of discretion required, set more realistic performance goals for the unit if discretion is not provided, or live with the fact that lack of discretion is hurting the unit's performance.

tion to accommodate, react, and resolve this uncertainty." (See Majchrzak [1997] for a discussion of the scientific literatures used in development of TOP Modeler.)

Initially, then, we represented the TOP Modeler knowledge-base as a set of over 1,500 if-then decision rules. To invoke a rule, a user provided company-specific knowledge about inputs (the "if" part of the if-then clause). This rule-based way of representing the knowledge had excellent predictive ability in tests comparing computer-based predictions to those of human experts. In addition, the other knowledge-based software package for organization design (Baligh et al. 1996) used a rule-based knowledge representation. Finally, academic experts hired to review the knowledge-base found that the rule-based format greatly facilitated evaluation.

While the rule-based representation worked for experts, observation showed it did not match *non-experts'* representations of organization design knowledge. Practitioners did not follow a set of rules to arrive at a single prescribed solution. Instead, they compared and traded off alternative solutions and eventually arrived at hybrid combinations that could not easily be traced back to a finite set of rules. Sometimes they proposed a solution and traced its probable impacts; sometimes they started from company constraints to see what solutions were possible; and sometimes they balanced many variables simultaneously and searched for an optimal solution in a way reminiscent of Pava's (1983) description of deliberations. This observation suggested to us that lay users represent their knowledge as *tradeoffs for action*, rather than the if-then rules of experts. The need for a way to translate expert knowledge

Exhibit 5. How the Development Process Used Functional Prototypes in Radical Iteration

Every Friday, Domain Team members downloaded a fully functional prototype from an FTP site and observed as someone in their organization worked with TOP Modeler to conduct a real use case of organization design analysis (an activity that took several hours). The Domain Team members reported back to us early the following week on how the analysis had gone. (How did the user use the system? What were the stumbling blocks, questions raised, or results that the user found of immediate relevance?) From this feedback, we were able to determine which aspects of the system were more likely to meet the needs of the unpredictable user community.

As an example, one Domain Team member (from her company's corporate engineering staff) downloaded the software on Friday and flew to a plant on Monday to meet with the plant manager and plant engineering staff for an organizational analysis. Upon her return, she reported to the Development Team the questions and frustrations of the plant personnel. Based on her input, changes were made to the analysis formulae. Another Domain Team member made the system available to a shop floor supervisor on Friday. The following Monday, we were informed that the supervisor had worked over the weekend, performed an organizational analysis of her production operations, and presented the results to management!

into actionable tradeoffs for non-experts led to development of the tradeoff matrices knowledge representation. Exhibit 4 explains how tradeoffs were represented and used in TOP Modeler.

Again, this shift in system design corresponded to a shift in our system development approach. Our *radical iteration* approach differed from the traditional prototyping approach in several ways. First, *functional* prototypes were used. Initially, as recommended by many decision support experts (Sprague and Carlson 1982; Watson et al. 1997), we interviewed potential users and encouraged their feedback on nonfunctional, throwaway prototypes of screens, reports, and interfaces. We found, however, that users' feedback on our early nonfunctional prototypes was of limited value. The task of organization design had never before been supported by a system. Users could not hypothesize how they might use an organization design support system from the limited evidence of simulated system features.[2] Second, in our radical iteration approach, users evaluated functional prototypes by working with the system through *real "use cases"* of organization design analysis, rather than hypothetical ones. Third, we continued iterating *far more times* than is customary with prototyping. During an 18-month period, over 70 functional prototypes were generated. (This activity was supported by the system's component-based architecture, described below.)

Each prototype was fully functional and specifically designed to shed light on some aspect of IT support for organization design. We tested alternative interfaces, different representations of the knowledge-base, multiple gap analysis formulae, different ways of providing method guidance, and alternative explanation styles, among other system features. Exhibit 5 provides examples of how the development process worked.

In sum, we learned that, when designing for emergent knowledge processes, asking people to review nonfunctional prototypes or participate in conference room pilots or "organizational games" is not sufficient to guide development, because these techniques involve hypothetical contexts. People infrequently perform EKPs, and they don't

[2]Zmud et al. (1993) have similarly commented on the difficulty of IS design when users must project into a hypothetical future.

really know how they might act when using a nonfunctional hypothetical support system; they can only demonstrate how they actually do act when confronted with a working system and a real use case. Using functional prototypes allowed us to intervene directly in the work process and observe which aspects of the system worked and which did not. Further, our radical iteration strategy was suited to the novelty of providing IT support for the emergent organization design process.

Principle #3: Design for *Offline* Action

When we first began observing users working with functional prototypes, users reported that TOP Modeler's tradeoff analyses were useful. But when we observed users *offline* (in organization design discussions in their organizations), we found they rarely acted on the information generated in TOP Modeler analyses. Overwhelmed by the sheer number of organization design options TOP Modeler considers (each with costs, benefits, interdependencies, feasibilities), users did not know what actions to take, and as a result they ignored TOP Modeler output when they made organization design decisions.

In this way, we learned that it was not enough to induce naïve users to try TOP Modeler (Principle #1) and to translate expert knowledge into actionable tradeoffs for lay users (Principle #2), we also had to induce naïve users to do things differently—to *act* offline on the basis of organization design knowledge generated online. Our observations of meetings in which organization design decisions were made revealed that lay organization designers spent a great deal of time trying to prioritize actions that would reduce the gaps between existing organization structures and desired organization performance measures. Should we provide training first, or should we invest in new equipment and do training later? Is it more important to initiate cultural change now, or wait until we've fixed our basic production problems? Therefore, to induce offline behavior change, we needed to provide support in TOP Modeler for action prioritization.

We initially expected that a single action prioritization approach would work for all potential users. But just as lay organization designers have an extremely wide range of relevant background knowledge, they also prioritize actions in very different ways. Some people prioritized actions by focusing on the biggest performance gaps first. Others focused on gaps related to their highest priority business objectives. Still others focused on gaps that could only be resolved with a particular solution. Therefore, providing several ways for users to prioritize actions to close gaps was necessary to our goal of influencing users' offline actions. Exhibit 6 indicates how TOP Modeler fulfilled this design principle.

Our recognition of the need for this design principle accompanied a major shift in how we conceptualized the system development process. Our initial view of the development process was formalized in the contract with NCMS and the partner companies. The contract spelled out our responsibilities to supply *a system* that would facilitate non-experts' use of expert organizational design knowledge. We came to realize, however, that we needed to supply *the system-facilitated application of* expert organization design knowledge to real organization design problems. If TOP Modelers' users did not actually take offline action, the goal of better organizational design decisions would not be achieved. Thus, instead of evaluating *our* performance in terms of what the system did and what users did when working with the system, we began evaluating our performance by what users did offline. Exhibit 7 describes how the development approach focused on offline action.

Principle #4: Integrate Expert Knowledge with Local Knowledge Sharing

We initially thought of organization design as a holistic organization-wide decision that could be informed by expert knowledge. Because improving organizational effectiveness requires coordinated inputs from human resource specialists, industrial engineers, manufacturing technology specialists, shop floor workers, supervisors, etc., as well as expert knowledge, we believed that an organization design support system should synthesize

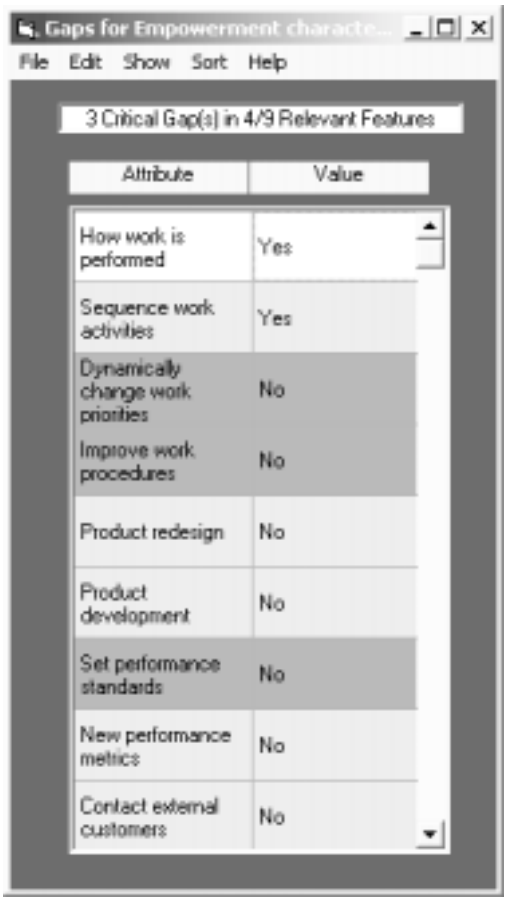


Figure 3. Summary Gap Panel

Exhibit 6. How TOP Modeler Facilitates Offline Action

TOP Modeler allowed users to prioritize their offline actions in three different ways. First, for those users who prioritized gaps by focusing on the worst cases first, the color-coded thermometers of the Ferris Wheel were helpful, because they indicated where the biggest gaps were located (areas where there were the most red thermometers).

Second, for those who focused on gaps related to their highest priority business objectives, we provided a "Summary Gap" panel that color-codes gaps based on the number of business objectives they affect. The example depicted Figure 3 involved production technology that did not allow shop floor operators discretion in three areas: dynamically changing work priorities, improving work procedures, and setting performance standards. As indicated at the top of the screen, only four out of the nine possible areas of discretion were relevant given the manufacturing unit's business strategies. Therefore, the fact that three out of four relevant areas of discretion were negative for the selected business strategies gives this gap a higher priority than it might otherwise have had.

The third way users prioritized gaps was by attending to "show stoppers"—gaps that could not be eliminated by making compensatory organizational changes. For these users, the Design Model was developed. (Figure 4 shows the Design Model.) If, for example, the workforce of a unit is missing a necessary skill (such as "repair"), the model indicates that the skill might be provided by making available someone who has the skill (a repairman); thus, this missing skill is not a show stopper and does not have to be fixed immediately through training. By contrast, as shown in Figure 4, failing to reward workers for the right outputs is rarely compensated by other features, and thus this gap *is* a show stopper.

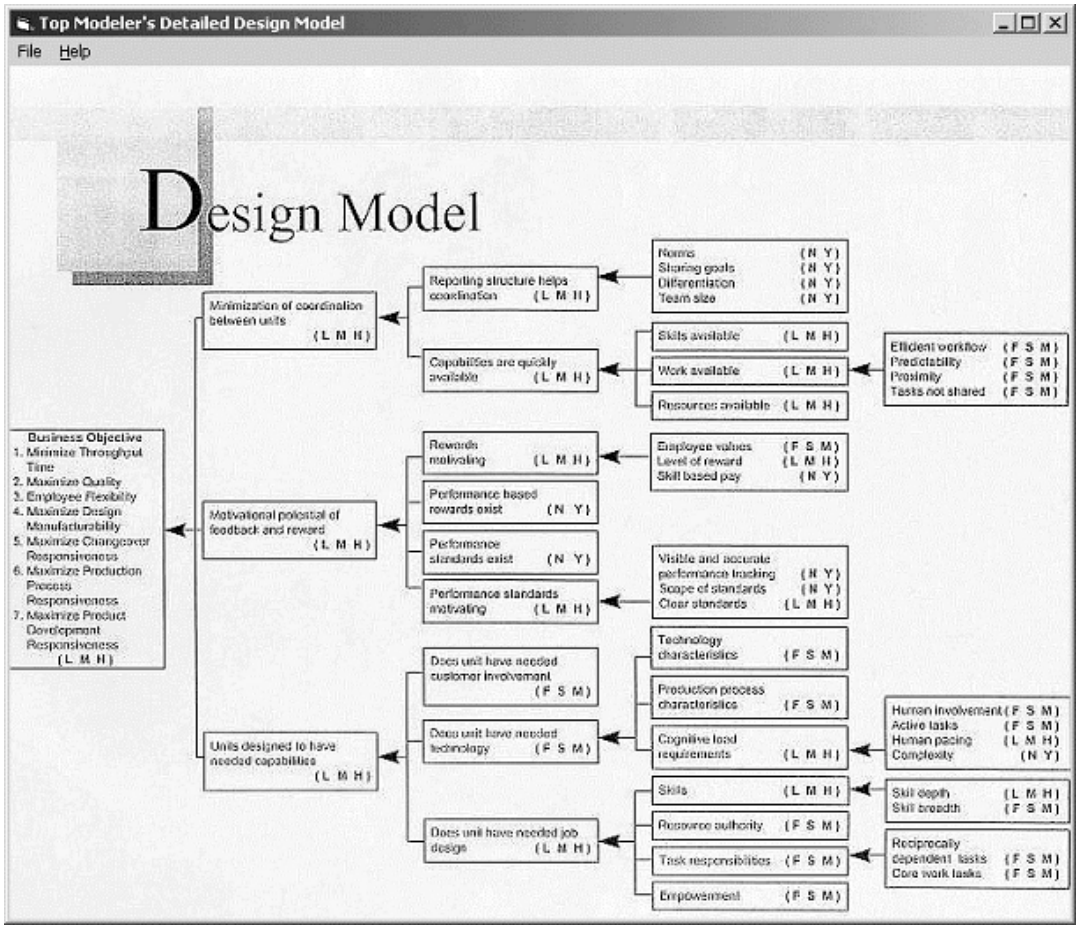


Figure 4. Design Model

Exhibit 7. How the Development Process Focused on Offline Action

We stopped asking users: What did you learn when you used the system? Why did you press those buttons? What will you do with that information? Instead, we started observing what they actually did in their organizations after they used TOP Modeler. We sent Domain team members and developers to watch design meetings before, during, and after TOP Modeler usage. By focusing on offline behavior change, the development process took many unexpected turns. One example was our learning the three different ways people prioritized TOP Modeler's gaps. Another example involved our recognizing a potential new user group for the system: maintenance workers. Observing maintenance workers using TOP Modeler indicated many translation problems between the manufacturing language and the maintenance language. This caused the Development Team to consider how the language in the system could be generalized beyond the manufacturing environment. We then made changes in the knowledge-base to broaden definitions and facilitate translation.

expert and diverse local knowledge inputs into a single, consensus perspective. We believed that the components of a traditional expert support system—knowledge-base, inference engine, and interface—would promote such a synthesis.

Over the course of the project, we learned that users had very different perspectives on organization design, shaped by their positions and job requirements. Shop floor workers did not share the concepts or the concerns, let along the knowledge, of manufacturing engineers, yet the input of all is needed for a good organization design. Unfortunately, we observed that functional prototype users rarely felt the need to consider others' points of view. Despite their limited understanding of many organization design issues, they did not solicit inputs from people in other functional areas. Further, they interpreted the system's knowledge-base through the lenses of their functional specialties. As a result, they performed incomplete analyses and produced ineffective solutions to organization design problems.

Consequently, we recognized that the components of a traditional expert support system—knowledge-base, inference engine, and interface—were not enough. We had to integrate the expert knowledge-base with system design features that might promote knowledge sharing among organizational members in different functional areas. In this way, we came to see that successful emergent knowledge support systems must represent a fusion of multiple "system types." They are not just decision support or expert systems but also knowledge sharing systems. We also learned that unstructured communication systems are not the only effective way to support emergent knowledge processes: expert knowledge repositories combined with knowledge sharing features are a good solution, too. Exhibit 8 describes the features of TOP Modeler that supported local knowledge sharing.

In keeping with this design principle, the development process changed. Initially, most of our focus was on translating expert knowledge into actionable knowledge for users. As we increasingly focused on users' offline behavior and observed their lack of interest in others' local knowledge, we began designing to influence their offline knowledge sharing as well as their prioritization of actions to resolve gaps. We observed who users communicated with before and after using TOP Modeler and who they involved in organization design deliberations.

Principle #5: Design for Implicit Guidance Through a Dialectical Development Process

We initially assumed that the way to support lay organization designers with TOP Modeler was to

Exhibit 8. How TOP Modeler Encourages Local Knowledge Sharing

We redefined the terms in the knowledge-base more generally so that people from different functional areas could make locally relevant interpretations. We tried to increase opportunities for shop floor workers and supervisors to participate in organization design decisions by building into the knowledge-base the concepts most relevant to them—workers' skills, work norms, and responsibilities for 144 specific shop floor tasks. We added annotation capabilities that made it possible for users to document inputs and analyses. We added the capability to save and e-mail analysis results to others for evaluation and as a catalyst for discussion. We added the capability to save partial inputs into a case so that, for example, managers could suggest business objectives and ask workers to complete a redesign that met those objectives. We provided full-screen visuals for projection to promote the use of the system and its analyses in meetings. And we redesigned the system technically so that it could be easily deployed on multiple platforms, whether on the shop floor or in the boardroom.

guide (Silver, 1991) them explicitly through the process that expert organization designers use. The envisioned process included the following steps: objectively conduct a compete socio-technical analysis, submit the analysis to co-workers for deliberation, evaluate the effects of each organizational dimension on business strategies and on other dimensions, and collectively decide on a single course of action. To support this explicit process, we constructed a "roadmap" metaphor for TOP Modeler's interface design.

However, we found the principle of explicit guidance faulty in two respects. First, expert organization designers did not themselves follow such a roadmap. Because of the emergent nature of organizational design, expert organization designers need process flexibility. Second, explicit process guidance did not work with non-experts. Some TOP Modeler users deliberately circumvented the roadmap and refused to conduct a thorough sociotechnical analysis. For example, one manager was observed to use the system to *prove* that his organization was already properly structured by focusing on only one aspect of the knowledge-base, when a more comprehensive analysis would have revealed numerous gaps. Further, since the system at that time did not promote knowledge sharing across functional areas, there was no self-correcting mechanism for users' failures to do a good analysis. As designers, we viewed this and similar examples as evidence of design failure, and we sought a better solution.

Eventually, we realized that the autonomy of knowledge workers makes explicit process guidance risky and failure-prone. We had no way to ensure that they would conduct complete analyses or engage their co-workers in deliberations about the meanings of terms, interpretations of findings, and evaluations of alternative actions. So, instead of guiding users explicitly, we guided them implicitly. A key design decision was to replace the roadmap with the Ferris Wheel as the interface metaphor, thereby encouraging fuller analysis. To promote deliberations, we added extensive explanations. How TOP Modeler implicitly guided users toward fuller sociotechnical analyses and deliberations with co-workers is explained in Exhibit 9.

Arriving at the Ferris Wheel interface required us to change the way we dealt with conflicting requirements, such as the conflict between non-experts' need for guidance in organization design and their undeniable autonomy in system use. We initially pushed for consensus, as recommended in the 50 IS development textbooks reviewed by Salzman and Rosenthal (1994). However, we found that pushing for consensus sharply limited what people would be able to do with the system. We then tried a principle of providing "both-and capabilities" that gave the appearance of reconciling the conflicting requirements. For example, we developed a button that switched between the roadmap and a "quick-start" option. But this solution did not meet our objec-

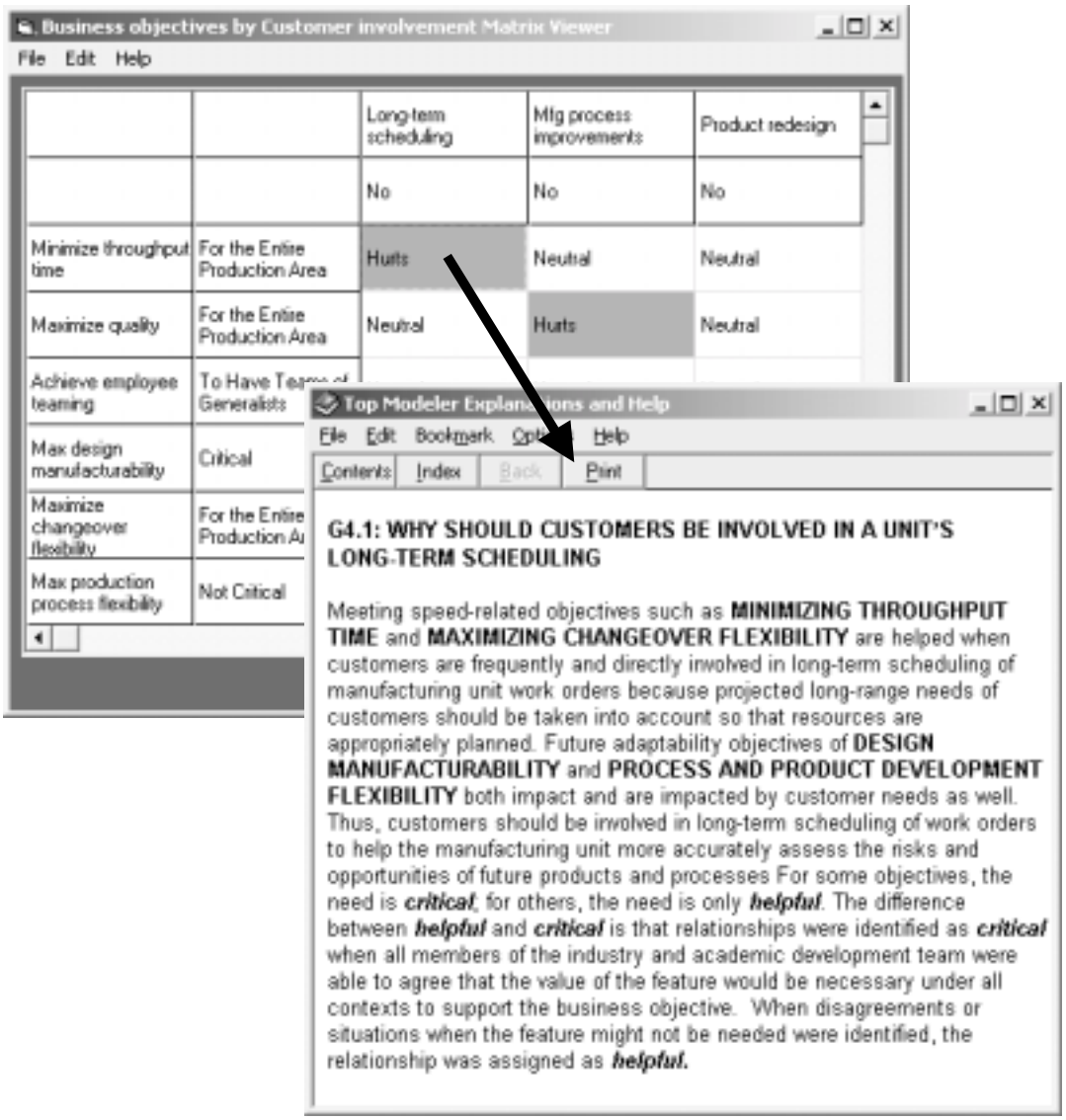


Figure 5. Example Explanation

Exhibit 9. How TOP Modeler Guides Users Implicitly

To guide users to conduct more complete sociotechnical analyses, we created the Ferris Wheel as the initial screen and main interface to TOP Modeler. In the center of the Ferris Wheel were business strategies, surrounded by the 12 sets of organizational design features. We intended this representation to imply that all 12 organizational design features must be aligned with business strategy; therefore, one should not selectively attend to one feature and ignore the others. User testing revealed that the Ferris Wheel interface did indeed accomplish the goal of conveying an intuitive grasp of the need for comprehensive sociotechnical analyses.

Further, because we set default values on the Ferris Wheel to zero, the thermometers on the Wheel initially showed red. This motivated users to continue their analysis until they could resolve the worst organizational gaps (the red thermometers). For example, Figure 1 presents results from a unit that was generally following best practices to achieve the unit's business strategies. However, the thermometers (dark gray in the figure, but red on the screen) clearly indicate inadequate involvement of customers. The businessman confronted with this evidence of under-performance was challenged to take action offline to improve the situation.

To encourage deliberations among co-workers, we precisely defined each term in a way that required information from people knowledgeable about the particular organizational design feature. For example, TOP Modeler posed questions about norms (behavior expected of employees) encouraged by management. Management encouragement of a norm was defined in terms of: mangers' verbal and operational sanctions, managers' demonstrations of the norm, and specific examples of norm compliance observable on the last shift. This definition encouraged the user to seek out shop floor workers to determine if specific examples of a particular norm had been demonstrated recently. As a result, they were often drawn into deliberations.

In addition to detailed definitions of terms, explanations for every result were provided. Figure 5 shows an example of an explanation. These explanations were specifically designed to encourage discussion and deliberations.

tives for guiding users' offline actions. In the end, we adopted a dialectical approach to development (Churchman 1979; Truex et al. 1999) that enabled a more fundamental resolution of the conflicting requirements.

In a dialectical development process, contradictions in requirements are not viewed as hurdles to be overcome. They are actually the *mechanism* by which effective systems support for EKPs is created. The emergent design process is one in which the designer responds creatively to the unexpected failures of a prototyped design feature to achieve its desired effect. Exhibit 10 describes the dialectical nature of our development process.

Principle #6: Componentize Everything, Including the Knowledge-base

From the outset, we had planned a component-based architecture that would isolate the knowledge-base from the inference engine and interface. Componentization has long been recommended by DSS and ES developers as an aid to software construction and maintenance (Nikolopoulos 1997; Sprague and Carlson 1982).

What we had *not* anticipated was the *extent* of componentization required in TOP Modeler. Because the organization design process was emergent, we continually encountered new users with new use cases, thwarting every attempt we

Exhibit 10. How the Development Process Was Dialectical

The dialectical approach used in the development process for TOP Modeler consisted of several steps. First, we clearly articulated apparent contradictions between user requirements as dilemmas, with valid arguments in support of each side. We then selected one side of the dilemma, developed an approach for effectively satisfying the needs epitomized by that side, and then created a prototype exemplifying the approach. To assess how successful we had been, we used criteria appropriate to that side of the dilemma. The roadmaps are an example of a solution for the "provide guidance to non-expert users" side of the dilemma.

Next, we went through the same process for the other side of the dilemma. For example, we developed an interface that provided essentially no guidance by allowing users to click through the knowledge-base as they saw fit. We then identified the underlying regularity, structure, or domain content that was common to the solutions for both sides of the dilemma and incorporated both the solutions and the commonality into the final product. Our creation of the Ferris Wheel interface illustrates the results of such a dialectical approach.

Exhibit 11. How TOP Modeler Demonstrates Radical Componentization

Within the interface component, we had the following components: a matrix viewer (used with the Tradeoff Matrices discussed above), a gap viewer, the Ferris Wheel viewer, roadmaps, and a Design Model viewer. Within the inference engine, we had the following components: a constraint satisfaction system and a "gap" aggregator (for computing aggregated values that appeared in the Ferris Wheel thermometers). Within the knowledge-base, we had the following components: the Tradeoff Matrices, the more complex Design Model, definitions of terms and concepts, and explanations. Even the Tradeoff Matrices had components, corresponding to the 12 sets of organizational features for each business strategy. Finally, system explanations were componentized as well. For every object in TOP Modeler—every screen, every organizational feature, every gap result—there was a componentized explanation composed of three parts: definition, functional explanation, and contextual explanation. In all, there were over 21,000 explanations. Figure 6 shows a schematic view of TOP Modeler's component-based architecture.

made to stabilize the knowledge, inference engine, and interface. In reaction, we proceeded to componentize virtually every aspect of the system. Exhibit 11 demonstrates the extent of componentization in TOP Modeler.

The componentized architecture ensured that, as domain knowledge evolved, the system would evolve with it. As components were modified, they could be dynamically "plugged into" the generic system structure to very quickly create a new, testable system for user evaluations. This architecture allowed us to create and experiment with over 70 significantly different system versions in a period of 18 months—a pace of innovation not commonly found in ES, DSS, or EIS development (Watson et al. 1997).

In addition, the componentized structure allowed for easy post-development modifications to the system. For example, changing explanations of concepts and relationships was easily done by a designated knowledge-base maintenance person: changes are made in a Word document and the

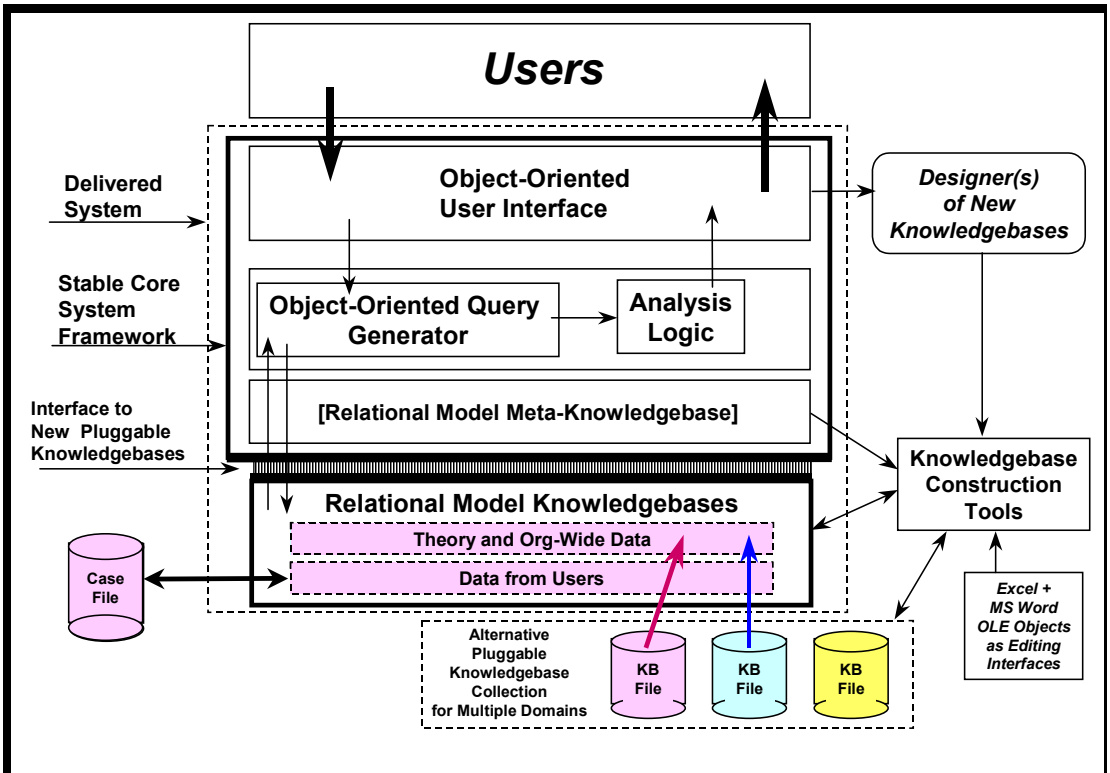


Figure 6. Top Modeler System Architecture

system is recompiled with the new Word document. Similarly, changes to the matrices can be made by simply updating an Excel spreadsheet.[3]

Our strategy of radical componentization in the design of TOP Modeler had significant effects on the development process. Much of the time in development team meetings was spent on working out critical assumptions about how the components would work together. Often, the assumptions only became clear when they were violated by experience. For example, the developer of the matrix viewer assumed a certain regularity in the database that was not initially intended, but was initially present. As development proceeded, that regularity was removed, and problems with the matrix viewer arose. Much discussion ensued: Should the matrix viewer accommodate lack of regularity in the database, or should the database be made more regular? To surface such issues as soon as possible, we instituted at least weekly integration builds, a practice employed in numerous open-source development projects and later popularized by Microsoft (Cusumano and Selby 1995; DiBona et al. 1999; Raymond 1999).

Radical componentization also allowed radical responses to changes in user requirements as our knowledge about IT support for EKPs grew. For example, two-thirds of the way through the development effort, the radical componentization in TOP Modeler made it possible for us to replace the roadmap interface with the Ferris Wheel. This system modification was so major that the Executive Steering Committee discouraged it,

[3]Since changes are so easy to make in TOP Modeler, a major concern is that the scientific grounding of the knowledge-base can be lost. Thus, a user group is generally organized in a company to monitor recommended changes to the knowledge-base.

believing that it would be too difficult to implement on time. The fact that the change *was* completed on time and within budget attests to the robustness of the system architecture and the flexibility of our development process.

In sum, the emergent nature of EKPs ensures unexpected problems and opportunities throughout the system development effort. Therefore, structuring the system and the development effort to pursue the unforeseen is a way to incorporate great ideas into the system design—even in the project's waning hours.

Contribution of EKP Design Theory

To recapitulate, we identified a class of design problems we call emergent knowledge processes. This class of problems has different process, user, and knowledge requirements from those of semi-structured decision-supporting systems. In addition, this class of problems is not adequately supported by existing system types (e.g., DSS, EIS, ES, groupware, etc.) either alone or in unintegrated combination. Therefore, a new IS design theory for systems that support EKPs is needed. Such a design theory matches principles guiding the selection of system features and principles guiding the development process with the unique user requirements of EKPs. Figure 7 summarizes our EKP design theory.

We argue that this EKP design theory is an important theoretical contribution, first, because it addresses the IT support and development process requirements *of an important class of human activities*. In addition to organization design, EKPs include the important processes of strategic planning, new product development, basic research, and indeed academic theory building (Weick 1989). More instances of this class may eventually be identified. An example is intellectual capital management (Stewart 1997). By showing how one process in the class can be successfully supported with IT, we pave the way for further attempts to develop IT support for EKPs. A plausible example is the application of knowledge mining technologies to the development of academic theory.

Second, our EKP design theory is an important theoretical contribution, because it shows how the *features* of familiar system types (such as DSS, EIS, ESS, etc.) *can be effectively integrated* (not just added) to accomplish effective support. Through its emphasis on integrated support, our EKP design theory helps resolve the considerable (and to our mind unproductive) disagreement in the knowledge management field (Fahey and Prusak 1998) about whether the best approach to supporting EKPs is via a high-tech "contentful" system, such as a case-based reasoning technology (El-Sawy and Bowles 1997), or via a low-tech communication-type system (i.e., user supplied content), such as video conferencing or email (Brown and Duguid 1998).

Third, our EKP design theory is an important theoretical contribution, because it shows how IS *development practices* need to be modified for the special requirements of EKPs. For example, a review of over 50 IS development textbooks (Salzman and Rosenthal 1994) articulates the development principle of striving for consensus among user requirements. How can this principle be applied to EKPs when specific user roles cannot be targeted? In such a situation, Salzman and Rosenthal recommend that, "rather than trying to achieve consensus" (p. 184), developers need to conduct a tradeoff analysis to determine the real and legitimate needs of different parties. This recommendation suggests that the design goal is a single, privileged solution, to which other needs are subordinated. But the many different event triggers of EKPs permit no single approach to prevail. Instead, EKP design theory recommends dialectical development as a way to design system features that reconcile, rather than trade off, conflicting requirements.

Fourth, our new IS design theory for EKPs is an important theoretical contribution, because it *both provides guidance to developers and sets an agenda for academic research*. EKP design theory makes the development process more tractable

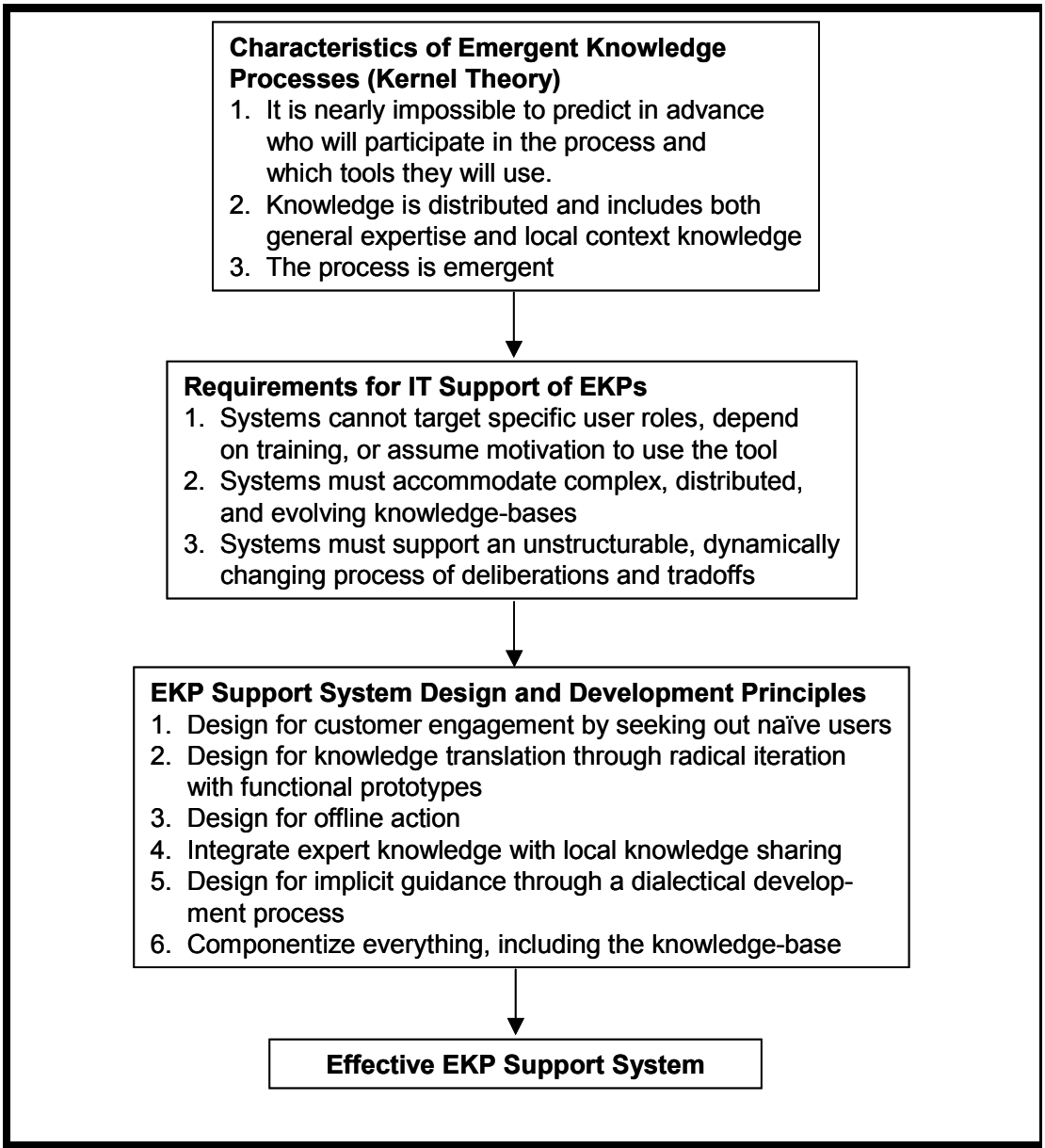


Figure 7. A Design Theory for Systems That Support Emergent Knowledge Processes

for developers by restricting the range of allowable features (or rules for selecting features) and development practices to a more manageable set. EKP design theory represents a total solution to the design problem by correlating user requirements (based on academic theories or practitioners' theories-in-use) with system design and development principles. Moreover, as an IS design *theory* (Walls et al. 1992), EKP design theory provides a set of *general principles* to solve *a class* of business problems, rather than *a unique set of system features* to solve *a unique* business problem. Thus, our design theory represents an advance from the largely atheoretical practice of requirements-driven software development.

EKP design theory also sets an agenda for academic research by articulating theory-based principles that are subject to empirical, as well as practical, validation. Just as Keen and Scott Morton (1978) made a contribution to the IS field by alerting researchers and developers to the existence of a new design situation that they labeled DSS, and just as Silver (1991) extended and formalized DSS design theory, so we too hope to make a contribution with our identification of the requirements, system features, and effective development practices of EKPs.

In the next section, we detail an agenda for research inspired by our theory. However, as our design theorizing was stimulated by our experience with a single revelatory case, we need to address the *generality* of our contribution. Because IS design theories are *theories* (Walls et al. 1992), our design principles can be restated as a set of hypotheses to be tested empirically in other situations where the same theoretical conditions hold (e.g., in work contexts characterized by emergent knowledge processes). In effect, our generalizability argument is that, because organization design has the characteristics of the *general class of emergent knowledge processes*, applying our EKP design and development principles to other specific EKPs (e.g., strategic planning) will result in new systems that successfully support those processes. Only the accumulated weight of empirical evidence will establish the validity of this hypothesis.

Conclusions

In this final section of our paper, we consider future research directions and practical implications.

Agenda for Future Research

As part of the *MIS Quarterly* themed issue on theory development, our conceptualization is only as good as its implications for further research. Two primary lines of future research flow from our design theory.

An obvious first step on the research agenda is to validate the design theory. Validation questions include:

- Do the requirements for IT support for EKPs outlined in Table 2 constitute a complete set? Are there alternative sets of requirements that also fit the kernel theory of EKPs?
- Can other development teams follow the EKP design and development principles to produce successful systems?
- Do systems that satisfy the principles outlined in Table 2 successfully enhance the outcomes of EKPs? In particular, is it possible to quantify the benefits of EKP support?
- Is it possible to demonstrate that the performance effects of using EKP support systems are really attributable to the systems instead of to "placebo" or "Hawthorne" effects?
- Are EKP support systems effective in all contexts? In the case of TOP Modeler, both Hewlett-Packard and General Motors were pilot sites. Despite the vast differences in organization structure and culture in these two organizations, the tool was successfully used in both. Does that suggest that organizational structure has limited effects on EKP support system use?
- Do the requirements and principles apply only to EKPs and EKP support systems, or can they also be usefully employed in the case of other

knowledge work processes (e.g., semi-structured decision making)? Put differently, is there any downside to providing *more* support for semi-structured decision-making than has proved successful in the past?

Beyond validation, the specific principles listed in Table 2 suggest additional challenging research issues.

- *The Principle of Designing for Customer Engagement by Seeking out Naïve Users*: How can an EKP support system be designed to encourage use, if users are not required even to try it? What kinds of novel system implementation strategies might be appropriate for EKPs?

- *The Principle of Designing for Knowledge Translation Through Radical Iteration with Functional Prototypes*: With EKPs there must be both deliberation and action. How can an EKP support system achieve an appropriate balance—not sending people off to make business changes willy-nilly, but not contributing to "analysis paralysis" either?

- *The Principle of Designing for Offline Action*: This principle raises difficult ethical issues for the system designer. What behaviors can and should one support? For instance, one client wanted TOP Modeler to determine how many people could be laid off in a manufacturing work setting. How far can and should the designer go to support business "needs"?

- *The Principle of Integrating Expert Knowledge with Local Knowledge Sharing*: Recent advances in text mining, directory standards, and "living portal" technologies suggest that much more can be done in the way of content-rich support for knowledge workers. However, crafting the right kinds of tools and integrating them with effective social interaction patterns will call for interdisciplinary research efforts involving computer scientists, anthropologists, and IS specialists.

- *The Principle of Implicit Guidance:* Tools like TOP Modeler send very strong signals to people about what they should do. This type of guidance can interfere with organizational authority systems. For example, in one of our pilot organizations, a shop floor supervisor obtained a copy of TOP Modeler, conducted an analysis of her organization, and reported to her management that her organization's performance was hindered by her lack of control over material handling resources. Management was taken aback at her boldness in making organization design proposals "outside of her purview" and promptly forbade her and all shop floor supervisors to use the tool. What responsibility do IT designers have to help organizations manage the unintended social and political consequences of using their tools?

- *The Principle of Radical Componentization:* This principle is intended in part to keep an EKP support system alive—updated, current, maintained. But who "owns" the job of keeping expert knowledge current? In-house experts who might dilute the scientific validity by serving local agendas? Or expensive third parties?

Finally, the overall EKP development strategy of *radical iteration and observation of naïve users working with functional prototypes on real use cases* also raises challenging research questions. Our development process involved weekly iterations with functional prototypes (70 in total) that were field tested by users in actual use cases under the careful observation of developers. This process required significant user involvement, far beyond the normal demands of the system development life cycle. In addition, it required the development team to place an independent observer into the client/user organization. The required resource commitments were clearly very high. While this approach has been used when piloting collaborative KMS for new product innovation (Majchrzak et al. 2000), the high resource requirements suggest the need for research on when such a development strategy is absolutely essential.

Implications for Practitioners

For practitioners, our EKP design theory has two major implications. First, the design theorizing

approach offers benefits over an atheoretical requirements-driven system development approach. With the feature list approach, developers are often faced with hard choices about which features to cut when time or budget runs short (Cusumano and Selby 1995). In addition, the traditional approach provides designers with little guidance about what to do if particular features fail to gain user acceptance or to produce desired offline outcomes. The design theorizing approach yields general principles to guide the system developer. Since there are always alternative features capable of satisfying general design principles, our approach allows the developer both guidance and flexibility.

Second, EKP design theory suggests that there is great potential scope for EKP support systems that combine both expert knowledge and local knowledge sharing. As of yet, no packaged applications or development tools for an integrated EKP support system exist—so such systems must be painfully developed from scratch in those areas most likely to yield strategic organizational payoffs. However, we believe our experience in the case of TOP Modeler—and the general principles we derived from it—provide development teams with guidance on where and how to start.

Acknowledgements

We would like to thank the organizers and participants of the Oklahoma Workshop in May 2000, as well as *MIS Quarterly* reviewers for their helpful advice on earlier versions. We would also like to thank the many people who made TOP Modeler possible.

References

Alavi, M. "Managing Organizational Knowledge," in *Framing the Domain of IT Management: Projecting the Future Through the Past*, R. W. Zmud (ed.), Pinnaflex Educational Resources, Cincinnati, OH, 2000.

Baligh, H. H., Burton, R. M., and Obel, B. "Organizational Consultant: Creating A Useable Theory for Organizational Design," *Management Science* (42:12), 1996, pp. 1648-1662.

Beyer, H., and Holtzenblatt, K. *Contextual Design: Defining Customer-Centered Systems*, Morgan Kaufmann Publishers, Inc., San Francisco, 1998.

Bhattacharya, S., Krishnan, V., and Mahajan, V. "Managing New Product Definition in Highly Dynamic Environments," *Management Science* (44:11), 1998, pp. 50-64.

Blair, D. C. "The Management of Information: Basic Distinctions," *Sloan Management Review* (26:1), 1984, pp. 13-23.

Boland, R., and Tenkasi, R. "Perspective Making and Perspective Taking in Communities of Knowing," *Organization Science* (6:4), 1995, pp. 350-372.

Borys, B., and Majchrzak, A. "User Concerns in Adapting Organization Design Theory to Organization Design Practice: Results of a Longitudinal Field Study" University of Southern California Technical Report, Los Angeles, 1999.

Brown, J. S., and Duguid, P. "Organizing Knowledge," *California Management Review* (40:3), 1998, pp. 90-111.

Cannon-Bowers, J. A., Salas, E., and Converse, S. "Shared Mental Models in Expert Team Decision Making," in *Individual and Group Decision Making*, J. J. Castellan (ed.), Earlbaum, Hillsdale, NJ, 1993.

Checkland, P. *Systems Thinking, Systems Practice*, Wiley, Chichester, 1984.

Cherns, A. B. "The Principles of Sociotechnical Design," *Human Relations* (29), 1976, pp. 783-792.

Cherns, A. B. "Principles of Sociotechnical Design Revisited," *Human Relations* (49), 1987, pp. 153-162.

Churchman, C. W. *The Systems Approach*, Dell, New York, 1979.

Clark, K. B., and Fujimoto, T. *Product Development Performance*, Harvard Business School Press, Boston, 1991.

Couger, J. D. *Creativity and Innovation in Information Systems Organizations*, International Thomson Publishing, New York, 1996.

Cross, N. "Descriptive Models of Creative Design: Application to an Example," *Design Studies* (18), 1997, pp. 427-440.

Cusumano, M. A., and Selby, R. W. *Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People*, Free Press, New York, 1995.

Davenport, T. H., DeLong, D. W., and Beers, M. C. "Successful Knowledge Management Projects," *Sloan Management Review*, 1998, pp 43-57.

Davenport, T. H., Jarvenpaa, S. L., and Beers, M. C. "Improving Knowledge Work Processes," *Sloan Management Review*, 1996, pp. 53-65.

DiBona, C., Ockman, S., and Stone, M. (eds.). *Voice from the Open Source Revolution,* O'Reilly and Associates, Cambridge, MA, 1999.

Durkin, J. "Expert System Development Tools," in *The Handbook of Applied Expert Systems*, J. Liebowitz (ed.), CRC Press, Boca Raton, FL, 1997.

El Sawy, O. A., and Bowles, G. "Redesigning the Customer Support Process for the Electronic Economy: Insights from Storage Dimensions," *MIS Quarterly* (21:4), December 1997, pp. 457-483.

Emery, M. *Participative Design for Participative Democracy*, Australian National University, Canberra, Australia, 1993.

Fahey, L. , and Prusak, L.. "The Eleven Deadliest Sins of Knowledge Management," *California Management Review* (40:3), Spring 1998, pp. 265-276.

Florman, S. C. *The Existential Pleasures of Engineering* (2nd ed.), St. Martin's Press, New York, 1994.

Frenkel, S. J., Korczynski, M., Shire, K. A., and Tam, M. *On the Front Line: Organization of Work in the Information Economy,* Cornell University Press, Ithaca, NY, 1999.

Galbraith, J. *Organization Design*, Addison-Wesley, Reading, MA, 1977.

Greenbaum, J., and Kyng, M. (eds.). *Design at Work: Cooperative Design of Computer Systems*, Earlbaum, Hillsdale, NJ, 1991.

Grudin, J. "Interactive Systems: Bridging the Gaps Between Developers and Users," *IEEE Computer* (24: 4), 1991a, pp. 59-69.

Grudin, J. "Systematic Sources of Suboptimal Interface Design in Large Product Development Organizations," *Human-Computer Interaction* (6:2), 1991b, pp. 47-196.

Guthrie, R., and Gray, P. "Junk Computing," *Information Systems Management* (13:1), 1996, pp. 23-28.

Hayward, S. "Knowledge Work Also Needs Design," Research Note SPA-11-1992, Gartner Group, Stamford, CT, July 2000.

Hutchins, E. "The Social Organization of Distributed Cognition," in *Perspectives on Social Shared Cognition*, L. B. Resnick, J. M. Levine, and S. D. Teasley (eds.), American Psychological Association, Washington, 1991.

Iansiti, M. *Technology Integration: Exploring the Interaction Between Applied Science and Product Development*, Harvard Business School, Cambridge, MA, 1992.

Keen, P. G., and Scott Morton, M. *Decision Support Systems: An Organizational Perspective*, Addison-Wesley, Reading, MA, 1978.

Kivijarvi, H., and Zmud, R. W. "DSS Implementation Activities, Problem Domain Characteristics, and DSS Success," *European Journal of Information Systems* (2:3), 1993, pp. 159-168.

Litterer, J. A., and Jelinek, M. "Design as a Setting for Useful Research," in *Producing Useful Knowledge for Organizations*, R. H. Kilmann (ed.), Jossey-Bass, San Francisco, 1983.

Majchrzak, A. "What To Do When You Don't Have It All: Toward a Theory of Sociotechnical Dependencies," *Human Relations* (50:5), 1997, pp. 535-565.

Majchrzak, A., and Finley, L. "A Practical Theory and Tool for Specifying Sociotechnical Requirements to Achieve Organizational Effectiveness," in *The Symbiosis of Work and Technology*, J. Benders, J. de Haan, and D. Bennett (eds.), Taylor and Francis, London, 1995, pp. 95-116.

Majchrzak, A., and Gasser, L. "TOP Modeler," *Information, Knowledge, Systems Management* (2:1), 2000, pp. 95-110.

Majchrzak, A., Rice, R. E., Malhotra, A., King, N., and Ba, S. "Technology Adaptation: The Case of a Computer-Supported Inter-Organizational Virtual Team," *MIS Quarterly* (24:4), December 2000, pp. 569-600.

Markus, M. L. "Toward a Theory of Knowledge Reuse: Types of Knowledge Reuse Situations and Factors in Reuse Success," *Journal of*

Management Information Systems (18:1), 2001, pp. 57-93.

Markus, M. L., and Keil, M. "If We Build It They Will Come: Designing Information Systems That Users Want To Use," *Sloan Management Review*, Summer 1994, pp. 11-25.

Mintzberg, H. *The Rise and Fall of Strategic Planning*, MacMillan, New York, 1994.

Mumford, E. *Effective System Design and Requirements Analysis*, Macmillan Press, London, 1995.

Nikolopoulos, C. *Expert Systems*, Marcel Dekker, New York, 1997.

Pava, C. *Managing New Office Technology: An Organizational Strategy*, Free Press, New York, 1983.

Poltrock, S. E., and Grudin, J. "Organizational Obstacles to Interface Design and Development: Two Participant Observer Studies," *ACM Transactions on Computer-Human Interaction* (1:1), 1994, pp. 52-80.

Raymond, E. S. *The Cathedral and the Bazaar: Musings on Linus and Open Source by an Accidental Revolutionary*, O'Reilly and Associates, San Francisco, 1999.

Rockart, J. F. "Engaging Top Management in Information Technology," Sloan Management Review, (25:4), 1984, pp. 3-17

Salzman, H., and Rosenthal, S. R. *Software by Design*, Oxford University Press, New York, 1994.

Sarker, S., and Lee, A. S. "Using a Positivist Case Research Methodology to Test Three Competing Theories-In-Use of Business Process Reengineering," *Journal of the AIS* (2:7), 2002 (online).

Shneiderman, B. "Codex, Memex, Genex: The Pursuit of Transformational Technologies," *International Journal of Human-Computer Interaction* (10:2), 1998, pp. 87-106.

Silver, M. S. *Systems that Support Decision Makers: Description and Analysis*, Wiley, New York, 1991.

Sprague, R. H., and Carlson, E. D. *Building Effective Decision Support Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Stein, E. W., and Vandenbosch, B. "Organizational Learning During Advanced System Development: Opportunities and Obstacles," *Journal of Management Information Systems* (13:2), 1996, pp. 115-136.

Stewart, T. A. *Intellectual Capital: The New Wealth of Organizations*, Doubleday, New York, 1997.

Taylor, J. C., and Felten, D. F. *Performance by Design,* Prentice-Hall, Englewood Cliffs, NJ, 1993.

Todd, P., and Benbasat, I. "The Impact of IT on Decision Making: A Cognitive Perspective," in *Framing the Domains of IT Management*, R. Zmud (ed.), Pinnaflex, Cincinnati, OH, 2000.

Trist, E., and Murray, H. (eds.). *The Social Engagement of Social Sciences, Volume II,* University of Pennsylvania Press, Philadelphia, 1993.

Truex, D. P., Baskerville, R., and Klein, H. "Growing Systems in Emergent Organizations," *Communications of the ACM* (42:8), 1999, pp. 117-123.

Vandenbosch, B., and Huff, S. L. "Searching and Scanning: How Executives Obtain Information from Executive Information Systems," *MIS Quarterly* (24:1), March 1997, pp. 81-99.

Vermesan, A. I. "Foundation and Application of Expert System Verification and Validation," in *The Handbook of Applied Expert Systems*, J. Liebowitz (ed.), CRC Press, Boca Raton, FL, 1997.

Walls, J. G., Widmeyer, G. R., and El Sawy, O. A. "Building an Information System Design Theory for Vigilant EIS," *Information Systems Research* (3:1), 1992, pp. 36-59.

Watson, H. J., Houdeshel, G., and Rainer, R. K. *Building Executive Information Systems and Other Decision Support Applications*, Wiley, New York, 1997.

Weick, K. E. *Sensemaking in Organizations*, Sage Publications, Thousand Oaks, CA, 1995.

Weick, K. E. "Theory Construction as Disciplined Imagination," *Academy of Management Review* (14), 1989, pp. 516-531.

Zmud, R., Anthony, W., and Stair, R. "The Use of Mental Imagery to Facilitate Information Identification in Requirements Analysis," *Journal of Management Information Systems* (9:4), 1993, pp. 175-191.

About the Authors

M. Lynne Markus is Trustee Professor of Management at Bentley College. Markus has over 20 years experience teaching, conducting research, and consulting in the information systems field. Her current research interests include the strategic positioning of electronic marketplaces, the challenges of B2B systems integration, and inter-organizational change management around B2B e-business initiatives. Recent publications include "Toward a Theory of Knowledge Reuse: Types of Knowledge Reuse Situations and Factors in Reuse Success" (*Journal of Management Information Systems*, 2001), "The Performance Impact of Quick Response and Strategic Alignment in Specialty Retailing" (*Information Systems Research*, 2000), and "What Makes a Virtual Organization Work—Lessons from the Open Source World" (*Sloan Management Review*, 2000). Markus holds a B.S. in Industrial Engineering from the University of Pittsburgh and a Ph.D. in Organizational Behavior from Case Western Reserve University. She is a member of the editorial board of *MISQ Executive* and a former member of the *MIS Quarterly* editorial board.

Ann Majchrzak is Professor of Information Systems at the Marshall School of Business at the University of Southern California. Her research interests focus on the design of computer-facilitated work environments, including the human, technology, and organizational context. Recent publications include "Radical Innovation Without Collocation: A Case Study at Boeing-Rocketdyne" (*MIS Quarterly*, 2001), winner of the SIM International Paper Award Competition, "Technology Adaptation: The Case of a Computer-Supported Inter-Organizational Virtual Team" (*MIS Quarterly*, 2000), winner of the *MIS Quarterly* Paper of the Year Award, and "Generating Testable STS Theory" (*Journal of Engineering and Technology Management*, 2001). She won the 2001 Best Paper Award for the Academy of Management Organizational Communication and Information Systems Division for a study of knowledge management and reuse in innovative environments.

Les Gasser is Associate Professor at the Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign. His major scientific interest has been to understand the technical and social aspects of organization-scale information processing, in both theory and practice. This work includes investigations of the nature of social-level knowledge, action, and learning; organization-scale knowledge-processing frameworks such as intelligent multi-agent systems; and concurrent object-based computing. Much of his practically oriented work has had a focus on manufacturing organizations and on issues of enterprise-integration for continuously learning, agile firms. He has published over 50 technical papers and five books in these areas. Gasser has been on the faculties of Computer Science, Systems Management, and Industrial Engineering at the University of Southern California, and has held visiting faculty posts at the University of Paris and the Ecole des Mines de Paris. He received his B.A. in English Literature, magna cum laude, from the University of Massachusetts in 1976, and his M.S. and Ph.D. degrees in Computer Science from the University of California, Irvine, in 1978 and 1984, respectively.