A Brief History Of Information System Design

Susan Gasson, College of Information Science & Technology, Drexel University

ABSTRACT

Information system design is often viewed as a stage in the system development life-cycle, concerned with the detailed "laying out" of system software - more akin to technical drawing than to design in an architectural sense. The intent of this paper is to retrieve the notion of design and to view design as an holistic activity, where form is conceptualized for a whole set of information system elements, some of which are physical and some abstract in nature. Every information system design process is unique, because every information system is embedded much more firmly in an organizational context and culture than physical artifacts. To manage this uniqueness, we need a more complex understanding of what design involves than that communicated by most IS texts.

The paper presents a review of design theories, derived from IS literatures and from other relevant literatures, such as organizational management and social cognition. Friedman and Cornford (1990) identify three phases of computer system development: (i) dominated by hardware constraints, (ii) dominated by software constraints, and (iii) dominated by user relations constraints. Evolution in conceptualizations of design are presented from these perspectives, then a fourth evolutionary stage is discussed: IS design dominated by business process constraints. This fourth perspective moves the design of information systems on from the limited perspectives offered by viewing an information system as synonymous with a computer system and resolves many of the theoretical conceptualization issues implicit in recent IS design writings.

At the conclusion, it is argued that current models of design focus on design *closure* and so de-legitimize the essential activities of investigating, negotiating and formulating requirements for an effective design. IS design faces five "problems" that need to be resolved: employing an effective model of design by with which to manage the labor process, defining the role of the information system, bounding the organizational locus of the system problem, understanding the cultural, social and business context of which the IS will be a part, and managing collaboration between cross-functional stakeholders. A dual-cycle model of design is proposed: one that focuses on "opening up" the design problem, as much as design closure. An understanding of this dialectic has significant implications for both research and practice of design; these are presented at the end of the paper.

KEYWORDS:

Information System Design, System Development Methods, Co-design of Business and IT Systems

1. THE DESIGN OF ORGANIZATIONAL INFORMATION SYSTEMS

Business organizations are increasingly moving their focus 'upstream' in the traditional, waterfall model of the system development life-cycle. Recent trends in information system implementation - standardization around a small number of hardware and software environments, the adoption of internet communication infrastructures, object-oriented and component-based software design, outsourcing and the use of customized software packages - have standardized and simplified the design and implementation of technical systems. Organizational information systems are no longer viewed as technical systems, but as organizational systems of human activity - business processes, information analysis and dissemination - that are supported by technology. Firms can therefore focus more on the strategic and organizational aspects of information systems, implementing cross-functional information systems that affect stakeholders from many different organizational and knowledge domains.

Yet existing approaches to information system design derive from a time when *technical* complexity was the core problem: so they are intended to bound and reduce the organizational 'problem' so that a technical system of hardware and software may be constructed. Existing design approaches treat complex organizational information systems as synonymous with information technology. They are based on models of individual, rather than group, problem-solving and cognition. We have few methods to enable stakeholders from multiple knowledge domains to participate in information system design. What methods exist are ad hoc and not based upon any coherent theoretical understanding of how collaborative design works. We have no models upon which to base future management approaches and methods for 'upstream' (from

the waterfall model of the traditional, technical system development life-cycle) information system design.

Winograd & Flores (1986) define design as “the interface between understanding and creation”. Unsurprisingly, given the difficulty of studying such a complex process, there are few models of design which are based upon empirical work, rather than theoretical conjecture or controlled experiments. Most models are also rooted in an individual perspective of design, rather than those group processes which occur in most IS design contexts.

As theories of design activity have evolved, so the definition of the term "design" itself has changed. In the information systems literature, design was initially viewed as the decompositional processes required to convert a structured IT system definition into a physical system of hardware and software. In the introduction to Winograd (1996), the author states:

> " Design is also an ambiguous word. Among its many meanings, there runs a common thread, linking the intent and activities of a designer to the results that are produced when a designed object is experienced in practice. Although there is a huge diversity among the design disciplines, we can find common concerns and principles that are applicable to the design of any object, whether it is a poster, a household appliance, or a housing development." *(Winograd, 1996, page v).*

Given these commonalities, we have to question why the design of an organizational information system is so much more problematic than the design of a physical artifact, such as a house. In the field of architecture, design has well-established principles and procedures, with established computer-based tools to support them. Yet information system design is often viewed as a single stage in a "structured" system development life-cycle, concerned with the detailed "laying out" of system software - more akin to technical drawing than to design in an architectural sense. The intent of this paper is to retrieve the notion of design and to view design as an holistic activity, where form is conceptualized for a whole set of information system elements, some of which are physical and some abstract in nature (for example, a particular approach to the management of organizational change, the physical information system's suitability for a particular group of users, or to its ability to provide a set of flexible organizational outcomes for a range of different stakeholder groups). For the purposes of this discussion, design is viewed as the process of conceptualizing, abstracting and implementing an organizational information system, rather than as a specific stage in the information system development life-cycle. Design is not viewed as giving form to system software, but as giving form to a whole set of information system elements, some of which are physical and some abstract in nature. The abstract elements may lead to such deliverables as a particular approach to the management of organizational change, the physical information system's suitability for a particular group of users, or to its ability to provide a set of flexible organizational outcomes for a range of different stakeholder groups. Every information system design process is unique, because every information system is embedded much more firmly in an organizational context and culture than physical artifacts. Abstraction and generalization are therefore much more complex than that required for a universal artifact that can be employed in many different concepts.

This paper draws on several literatures to derive an understanding of what design "in the round" means. Abstractions of design from the literatures on organizational theory, architectural and engineering design, human-computer interaction, computer-supported cooperative work, management information systems and social cognition are synthesized here, to present an holistic conceptualization of design as organizational problem-solving, individual and group activity, and management-oriented process models.

Friedman and Cornford (1990) identify three historical phases of computer system development and a putative fourth phase:

1) System development dominated by hardware constraints (mainly cost and reliability of hardware).

2) System development dominated by software constraints: developer productivity, expertise and team project management issues (dominated by meeting deadlines and project budgets).
3) System development dominated by user relations constraints: inadequate perception of user needs by developers and lack of prioritization of user needs.
4) *(Predicted)* Organization environment constraints.

This paper reexamines these phases from the perspective of their impact on design methods and paradigms. The fourth phase is redefined, in the light of technical developments, which have provided ubiquitous computing platforms and simplified the development of both intra- and inter-organizational information systems. Instead, it is argued that the fourth phase constraints are to do with business and IT system alignment. This is signified by the poor involvement of business managers and other non-IS people, leading to a poor understanding of boundary-spanning business strategy and application domain issues.

Design is most frequently equated with theories of decision-making and problem-solving in organizations. This paper analyzes the development of major theories in this area and traces their impact on the evolution of design approaches for organizational information systems. This evolution is shown in Figure 1. It is argued in this paper that, as the scope of organizational impact encompassed by information systems increased with time, so did the importance of integrating contextual and social requirements, for the system to operate.

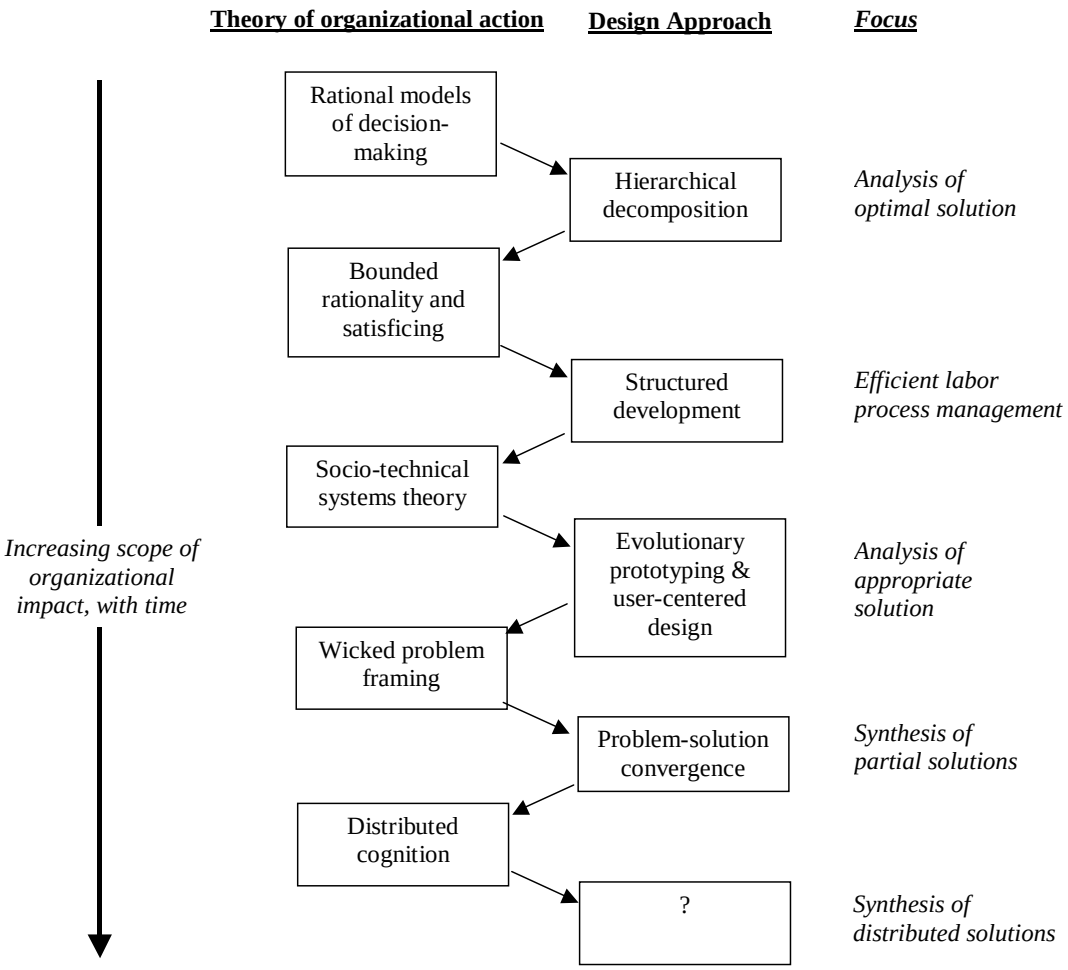

Figure 1: Evolution Of Design Methods Following Evolution Of Problem-Solving Models

The paper is structured as follows. Theories of design, as they relate to organizational information systems, are presented from the evolutionary perspective of the five theories of

decision-making. But theories of design do not follow a linear development of thought: they are interrelated and emergent. Relevant design theories are arranged according to the identified research *threads*, to reflect progressive and sometimes parallel developments in how design is perceived by the organizational and IS literatures. The use of Friedman and Cornford's (1990) "constraints" view of computer-system development permits a reasonable (if post-justified) perspective of why various theories were adopted by the IS community (both academic and practical) at a particular stage in the development of how IS were perceived. Section 2 deals with the evolution of early design theories: the attempt to apply, and modify, rational, "information processing" models to the development of systems limited by hardware constraints. Section 3 discusses the adoption of hierarchical decomposition and "structured" approached to design, as a way of dealing with software constraints. Section 4 presents the incorporation of theories of social construction, emancipation and "human-centered" design as a way of dealing with user-relations constraints. Section 5 presents the extension of design theory to encompass boundary-spanning information systems and discusses this extension as a response to strategic business coordination constraints. Section 6 summarizes the evolution of design theories, discusses some lacunae in current understandings of how design works and presents a dual-cycle model of design, to resolve some of the implications for design research and practice.

2. FROM RATIONAL DECISION-MAKING TO BOUNDED RATIONALITY

The concept of “rational” decision-making developed from Taylor’s (1911) “scientific management” principles and Weber’s (1922) "rationalization" of the social world. Both of these theories were concerned with optimization and quantifiable interpretations of natural phenomena, including human behavior. Simon’s (1945) book *Administrative Study* formalized rational decision-making into a linear, staged process model of intelligence-gathering, evaluation of alternative courses of action and choice. Early information system (IS) design theory stems from this perception of human behavior in organizations as rational decision-making. Human beings are seen as objective information processors, who make decisions rationally, by weighing the consequences of adopting each alternative course of action. One stage uses the outputs of the previous stage (hence the waterfall model of Royce, 1970). The information processing model of problem-solving (shown in Figure 1) assumes that all information pertaining to design requirements is available to the designer and that such information can be easily assimilated (Mayer, 1989).

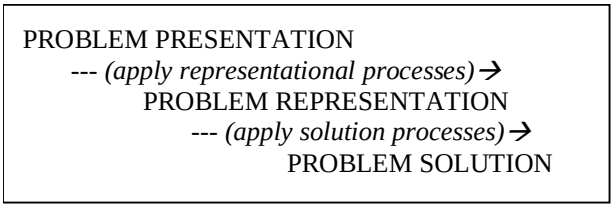

Figure 1: An Information-Processing Model of Problem-Solving (*after* Mayer, 1989)

In the information processing model, design involves moving from a statement of the problem in the world to an internal encoding of the problem in memory by mentally encoding the given state, goal states, and legal operators for a problem - i.e. by defining the problem mathematically. Solution involves filling in the gap between the given and goal states, devising and executing a plan for operating on the representation of the problem - i.e. by making a rational choice between alternative courses of action.

Two major advances in design theory modified, but did not replace, the notion of rational design. The first was that Alexander (1964) proposed that there is a *structural* correspondence between patterns embedded within a problem and the form of a designed solution to solve the problem. He proposed the process of hierarchical decomposition - breaking down the overall design

problem into a series of smaller problems (patterns) which can be solved independently from each other - as a way to accomplish complex problem-solving where all variables could not be grasped at once. The structure of an appropriate solution may then be determined mathematically, by analyzing interactions between the variables associated with the design problem (Alexander, 1964). As Lawson (1990) notes, this presupposes that the designer is capable of defining all the solution requirements in advance of designing the solution, that all requirements assume equal importance for the solution and that all requirements and interactions between them affect the *form* of the proposed solution equally. But there is a deeper consideration. Do external "patterns" exist, or do we simply impose them subjectively on external phenomena? Alexander himself (1966) criticizes the human tendency to artificially impose patterns on external elements that may constitute a design (in his example, he discusses architectural considerations affecting town planning). Yet there is a contradiction in Alexander's position: even his recent work assumes that patterns are somehow inherent in external "entities" and therefore that his method of pattern-matching may be employed to define objects for object-oriented system design (Alexander, 1999).

The second advance was that Simon (1960) introduced the principle of bounded rationality. Human-beings have cognitive limitations which constrain the amount of information they can absorb and that they have access to incomplete information about alternative courses of action. These limitations lead to high levels of uncertainty, to which humans respond by developing a simplified model of the real situation: they reduce, constrain and bound the problem until it becomes sufficiently well-defined to be resolved. Then they evaluate alternative solutions sequentially until an alternative is discovered which satisfies an implicit set of criteria for a satisfactory solution. The solution reached by this process of bounded rationality is not optimal, but *satisficing*, in that it satisfies a minimal, rather than optimal set of solution criteria.

This theory only appeared to apply to a subset of relatively well-defined design problems. Simon (1973, 1981) distinguished between well-structured and ill-structured problems. Well-structured problems may be resolved through the application of hierarchical decomposition techniques. But ill-structured problems (such as the design of a computer system) need to be structured before they can be analyzed. Individuals structure such problems by decomposing them into sub-problems: these are synthesized unconsciously so that the original, ill-structured problem "soon converts itself through evocation from memory into a well-structured problem" (Simon, 1973). The significance of this is worth noting: the process of problem structuring requires additional information, retrieved from long-term memory. This demonstrates a gradual realization that inductive reasoning (generalization from evidence) is significant in design. The “rational” model assumes deductive reasoning (logical inference about particulars that follows from general or universal premises). Even Alexander (1964) did not view the design process as entirely rational. In fact, he observes:

> “ Enormous resistance to the idea of systematic processes of design is coming from people who recognize correctly the importance of intuition, but then make a fetish of it which excludes the possibility of asking reasonable questions.” *(Alexander, 1964, page 9).*

By Simon's (1973) work, what Alexander refers to as "intuition" has become the application of inductive reasoning. However, a realization of the significance of inductive reasoning appears to lead to the notion that design is wholly "creative" in nature and therefore uncontrollable. Inductive reasoning involves conclusions drawn from particular cases in the individual's experience (the inference from particular to general), which is the antithesis of deductive reasoning, such as that involved in hierarchical decomposition. Thus, we come to the period of IS design dominated by pressures to manage the labor process.

3. FROM STRUCTURED DECOMPOSITION TO SOCIO-TECHNICAL SYSTEMS

The transition between constraint-phases was driven by the need to collaborate in the production of systems software. While IT systems remained relatively simple, a single designer could accomplish their execution. Once IT systems were used to support multiple organizational activities, it became necessary to employ teams of software designers. The evolution of design methods was driven by the need for collaboration and communication between individuals. Design approaches needed to develop a "common language". This was provided by the concept of structured design, based on Alexander's (1960) hierarchical decomposition.

Lawson (1990) presents a typical hierarchical decomposition model from the architecture field (Alexander's original work was in architecture), shown in Figure 2.

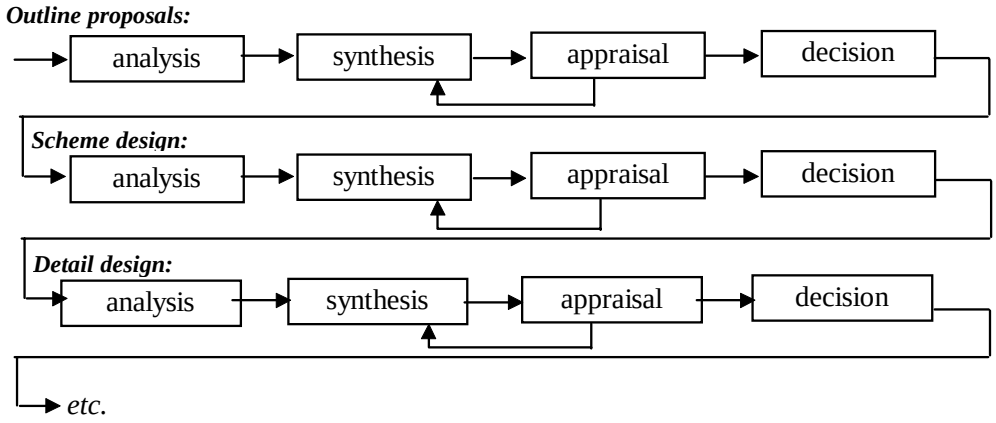

Figure 2: The Markus-Maver Model Of The Design Process *(after Lawson, 1990, page ???)*

Although the application of hierarchical decomposition appears first in the architectural design literature, this model was soon applied to IT system design. No single author can be credited with the development of the structured systems development life-cycle model that now underlies most IT systems development (the "waterfall" model), but the most influential presentation of this model is in Royce (1970). The model is so attractive because it seems to prescribe a way to control the labor process. By breaking the design process into stages (which are reproduced at multiple levels of decomposition), managers are enabled to (i) standardize work processes and (ii) divide design work between different people. This type of fragmentation in design-work distances people from the object of their work (leading to lower morale and productivity) and leads to uninformed decisions about design alternatives and form (Corbett et al., 1991).

The model is linear – while still popular in the IS field because of its focus on clearly-defined delivery milestones, this type of model has been rejected by many areas of creative design, such as architecture, as being unrepresentative of 'real-world' design processes (Lawson, 1990). McCracken and Jackson (1982) voiced the first dissent with structured, hierarchical decomposition in the IS literature, when they argued that this approach did not support the actual processes of design. They observed that IS professionals circumvented the method, then post-rationalized their designs by producing structured design documentation that made it look as if they had followed the method. But they concluded that the structured approach should be used anyway, because of the benefits for controlled and standardized process management:

> " System requirements cannot ever be stated fully in advance, not even in principle, because the user doesn't know them in advance … system development methodology must take into account that the user, and his or her needs and environment, change during the process." *(McCracken & Jackson 1982, p. 31)*

This paper argues for the *emergence* of system requirements through the process of design. But it is not until Boehm (1988) that structured approaches to design are seriously considered harmful to information system design because they ignore human-activity and task requirements of the information system:

" Document-driven standards have pushed many projects to write elaborate specifications of poorly understood user interfaces and decision support functions, followed by the design and development of large quantities of unusable code.'' *(Boehm 1988, p. 63).*

If we assess the empirical literature, studies of information system design tend to embody the assumptions of their contemporary theoretical literature. Earlier studies (e.g. Jeffries et al., 1981; Vitalari & Dickson, 1983) argue that failure is due to a lack of methodological consistency in applying structured decomposition. Later studies, with a wider scope of IS design in its organizational context (e.g. Jenkins et al., 1984; Curtis et al., 1988; Hornby et al., 1992; Davidson, 1993) argue that methodologies do not represent a 'theory-in-use', but a 'theory-of-action' (Argyris and Schön, 1978): they represent a rule-based interpretation of *what should be done*, rather than what people actually *do*. As time proceeds (with disciplinary knowledge), the role of *inductive* reasoning increases in importance. There is an appreciation of the role that tacit knowledge plays in design. The notion of "pattern-matching" from Alexander's (1964) early, positivist concept of deductive pattern-matching evolves to an inductive concept of convergence, involving the progressive fit of partial problem-definitions to partial elements of known solutions to such problem-patterns (Turner, 1987). Alexander himself critiqued his early notion of design as deductive variance-reduction. His recent work (e.g. Alexander, 1999) demonstrates a rich appreciation of design as inductive pattern-matching. In this sense, an individual design process as problem-solution convergence becomes very similar to the notion of design emergence discussed below.

4. DESIGN AS EXPERIENTIAL LEARNING

As information systems designers became more ambitious in the scope of organizational support that they attempted, information systems became more complex and the focus of design shifted to solving organizational and informational problems, rather than data processing. An evolution in thinking about organizational problems is demonstrated by Rittel (1972; Rittel and Webber, 1973) and Ackoff (1974). Ackoff (1974) described organizational problems as "messes", arguing that organizational problem selection and formulation are highly subjective:

" Successful problem solving requires finding the right solution to the right problem. We fail more often because we solve the wrong problem than because we get the wrong solution to the right problem." *(Ackoff, 1974, page 8)*

Rittel (1972; Rittel and Webber, 1973) suggested that organizational problem are "wicked" problems. A wicked problem has the following characteristics:

a) it is unique
b) it has no definitive formulation or boundary
c) there are no tests of solution correctness, as there are only 'better' or 'worse' (as distinct from right or wrong) solutions
d) there are many, often incompatible potential solutions
e) the problem is interrelated with many other problems: it can be seen as a symptom of another problem and its solution will formulate further problems.

Wicked problems (and messes) differ from Simon's (1973) ill-structured problems in one important respect. Ill-structured problems may be structured by the application of suitable decompositional analysis techniques: they may be *analyzed* (even if not rationally, in a way that may be justified on rational grounds). But wicked problems cannot even be formulated for analysis, because of their complexity and interrelatedness (Rittel & Webber, 1973). Rittel (1972) argued that such problems cannot be defined objectively, but are *framed* (selected, artificially bounded and defined subjectively and implicitly). Even once a wicked problem has been subjectively defined, the designer has no objective criteria for judging if it has been solved (in computing terms, there is no 'stopping rule'). Rittel advocated 'second-generation design methods' to replace the rational, decompositional model of design. These methods should

include “designing as an argumentative process”, which Rittel saw as “a counterplay of raising issues and dealing with them, which in turn raises new issues and so on”.

The realization that complex system design required experiential learning (Lewin, 1951) coincided with a demand-driven approach to IS design. As information systems expanded their scope, so information system users exerted their power. The rise of evolutionary prototyping was driven by their demands for more usable systems, but also by the fit with experiential learning. But it is a pragmatic fit, rather than an explanatory fit – the prototyping approach supports the need for experiential learning, but it does not explain the processes or behaviors of those people engaged in design. A convincing explanation is provided by Turner’s (1987) suggestion that design problems and solutions converge together. Information System design can be conceptualized as the progressive ‘fitting’ of the framework of system requirements that represent the problem with known solutions, based upon the designer’s previous experience of problems of a particular type (Turner, 1987). Turner observed various strategies employed by computer science and other students when resolving a semi-structured design task and concluded that goal definitions evolve with the design. Turner (1987) observed that, where designers’ own experience failed to provide a solution, they widened the search space to call on the experience of colleagues. Turner argued that “requirements and solutions migrate together towards convergence” and that the process of designing information systems is subjective as well as emergent:

> “ Design appears to be more ad hoc and intuitive than the literature would lead us to believe, solutions and problems are interrelated and the generation of solutions is an integral part of problem definition. Problems do not have only one solution; there may be many. Consequently, design completeness and closure cannot be well-defined. There are two categories of design factors: subjective and objective. Objective factors follow from the subjective concepts on which designers model the system. The difficulty in the past is that we have not acknowledged, explicitly, the presence of subjective factors, with the result that, in many cases, objective factors appear to be arbitrary.” (Turner, 1987, page ??).

We are then faced with the problem of how designers determine the “subjective concepts” on which they model their systems. In an empirical study of architects by Darke (1978), the author discovered that there was a tendency to structure design problems by exploring aspects of possible solutions and showed how designers tended to latch onto a relatively simple idea very early in the design process (for example, “we assumed a terrace would be the best way of doing it”). This idea, or ‘primary generator’ was used to narrow down the range of possible solutions; the designer was able to rapidly construct and analyze a mental archetype of the building scheme, which was then used as the basis for further requirements search. Darke’s (1978) model of the design process is shown in Figure 3.

Figure 3: Darke’s (1978) Model Of The Design Process

Darke’s (1978) architectural design model finds a parallel in the IS literature, in a protocol analysis study of information system design dialogues between designer and user (Malhotra et al., 1980). They highlighted the core role played by cognitive *breakdowns* (Winograd and Flores, 1982, *after* Heidegger, 1960) in making the implicit become explicit. They concluded that internal (mental) models held by designers often relied on assumptional, implied, rather than explicit requirements. These assumptions only surfaced when an implicitly-held requirement conflicted with an explicit user requirement, in dialog with system users. Designers often examined partially proposed design elements to test violation of an unstated goal and attempted to fit alternative solutions to subsets of the requirements, based on prior experience. Design goals *evolve* in with the learning that accrues from the process of design.

Turner (1987) concluded that, in practice, only some ambiguities of design requirements and goals will be resolved and the central issue becomes one of discrimination between the significant and the insignificant. Strategies for such discrimination have been linked with “opportunism” in studies of software design (Guindon, 1990a, 1990b; Khushalani et al., 1994).

Ball & Ormerod (1995) review the notion of opportunism in system design, which they define as deviation from hierarchical decomposition (top-down, breadth-first) design. They compare opportunistic design with the more structured problem-solving approaches observed in earlier studies of software design. They conclude that much of the structure observed in the early studies of design arose from the more structured nature of the problems set for subjects in experimental situations.

These ideas are synthesized in the diagrammatic model given in Figure 4. The design process is both iterative and recursive, redefining parts of the problem as well as partial solutions. In practice, only some ambiguities of design requirements and goals will be resolved and the central issue becomes one of discrimination between the significant and the insignificant (Turner, 1987).

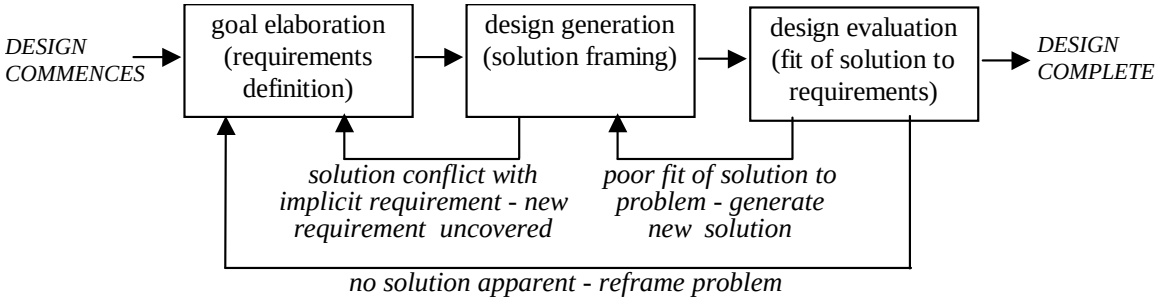

Figure 4: Proposed Model Of Individual Cognition in IS Design

It can be argued that the garbage-can model (Cohen et al., 1972) of problem-solving is only an explanation for observed phenomena that are too complex for the authors to explain. An alternative explanation could be that organizational actors do not search for problems to fit their solutions, but that problem and solution converge by a process of pattern-matching. Turner's (1987) convergence findings and Malhotra et al's (1980) conclusions about breakdowns would lead one to conclude that this type of problem-solving, especially in design, is emergent and contingent upon interaction with multiple perspectives of the problem-situation.

Curtis et al. (1988) conclude that "developing large software systems must be treated, at least in part, as a learning, communication and negotiation process." Designers have to integrate knowledge from several domains before they can function well. They identify the importance of designers with a high level of application domain knowledge: in their studies, these individuals were regarded by team members as "exceptional designers", who were adept at identifying unstated requirements, constraints, or exception conditions, possessed exceptional communication skills. Exceptional designers spent a great deal of their time communicating their vision of the system to other team members, and identified with the performance of their projects to the point where they suffered exceptional personal stress as a result. They dominated the team design process, often in the form of small coalitions, which "co-opted the design process". While these individuals were important for the *depth* of a design study, teams were important for exploring design decisions in *breadth* (ibid.).

This supports a perspective found in the literature on design *framing*: that decompositional approaches to system requirements analysis are of most use when designers are inexperienced, or when the design problem is unusually difficult to define (Jeffries et al., 1981; Turner, 1987). But the literature does not tell us whether the methodology is useful for supporting design activity in these situations, or whether its function is to provide psychological support in conditions of high uncertainty, as suggested by Reynolds & Wastell (1996).

Guindon (1990b) argues that information system design involves the integration of *multiple* knowledge domains: the application domain, software system architecture, computer science, software design methods, etc.. Each of these domains represents a problem-space in which a more or less guided search takes place (depending upon which solution paths look most

promising and the previous experience of the designer in this domain). The IS development process should encompass the discovery of new knowledge, in particular the discovery of unstated goals and evaluation criteria.

3.3 From Rational Problem-Solving To Problem-Framing

Human-Centered Design

The concept of the wicked (or socially constructed, multi-perspective problem) is reflected in the literature on participatory design, although this literature was driven also by an interest in worker emancipation and “human-centered” design. Research evidence indicated that the traditional approach to the development of new technology resulted in technological systems which were associated with a high degree of stress and low motivation among their users (Corbett, 1987; Gill, 1991; Scarbrough & Corbett, 1991; Zuboff, 1988). The human-centered approach to the design of technology arose as a reaction to this evidence. Gill (1991) defines human-centeredness as "a new technological tradition which places human need, skill, creativity and potentiality at the center of the activities of technological systems." Bjorn-Andersen (1988) criticized the narrow definition of human-computer interaction used by ergonomics and systems design research, which takes technology as its starting point, with the words: "it is essential that we see our field of investigation in a broader context. A 'human' is more than eye and finger movements". There is a wide body of literature on the development and application of human-centered technology. Some of the main ideas of this literature are:

1. The human-centered approach rejects the idea of the "one best way" of doing things (Taylor, 1947): that there is one culture or one way in which science and technology may be most effectively applied (Gill, 1991).
2. Technology is shaped by, and shapes in turn, social expectations: the form of technology is derived from the effect of these social expectations upon the design process (MacKenzie and Wajcman, 1985). This social constructivist approach reveals the *social* interior of technological design: technology no longer stands as an independent variable, but an outcome which is the result of socially-constrained choices made by designers.
3. The human-centered approach is opposed to the traditional, technically-oriented approach, which prioritizes machines and technically-mediated communications over humans and their communicative collaboration (Gill, 1991). While technically-oriented design traditions see humans as a source of error, the human-centered design approach sees humans as a source of error-correction (Rosenbrock, 1981).
4. That human-centered production should concern itself with the joint questions of "What can be produced?" and "What should be produced?" The first is about what is technically feasible, the second about what is socially desirable (Gill, 1991).
5. That objective and subjective knowledge cannot exist independently of each other: while technologists attempt to encode the explicit, rule-based knowledge needed to perform a task, this knowledge is useless without the "corona" of tacit and skill-based knowledge which surrounds the explicit core and through which explicit knowledge is filtered (Rosenbrock, 1988). Cooley (1987) raises the issue that modern technology is designed to separate "planning" tasks from "doing" tasks (for example, in modern Computer-Integrated Manufacturing). This results in deskilled human technology users, who are less equipped for exception-handling as a result, and poorer work outcomes, as those who plan are uninformed by seeing the results of their plans and those who "do" are unable to affect the way in which work tasks are approached.

A common theme in the human-centered literature is that it is the process of technology **design** which determines the effect of that technology upon its human users. This is best illustrated by

considering recent developments in the approach to technological determinism. Technology may be argued to determine work design (Braverman, 1974), or to be neutral in its impact, with the relationship between technology and work design being mediated by managerial intentions and values (Buchanan and Boddy, 1983), by managerial strategic choice (Child: 1972) or by organizational politics (Mumford & Pettigrew, 1975; Child, 1984). However, the *forms* of available technology have an independent influence on the range of social choices available (Wilkinson, 1983; Scarbrough & Corbett, 1991). An analysis of technology as an unexplored entity which simply embodies the intentions and interests of particular groups ignores the technological decision-making which *precedes* the managerial decision-making process: the processes of design.

This socio-technical perspective is most apparent in the literature analysis of prototyping and participatory design. This area of work explicitly attempts to deal with the "multiple worlds" espoused by various organizational actors (Checkland, 1981). Evolutionary methodologies permit users to incorporate desired ways of working into the design of the information system (Eason, 1982; Floyd, 1987). IS stakeholders are placed in a situation where they can negotiate their requirements of an IS around a design exemplar - a prototype IT system, or a prototype work-system. But the attempt to balance the two domains tends to focus more on one domain than the other. Whilst, for example, Mumford's work in ETHICS (Mumford, 1983; Mumford and Weir, 1979) attempts the joint satisfaction of both social and technical interests, it deals almost exclusively with the design of work systems. Technology is viewed as infinitely configurable to suit the organization of workgroups, with no account taken of constraints imposed by either technology design or its implementation. More recent work (Butler and Fitzgerald, 2001; Lehaney et al., 1999) examines the ways in which user participation in decisions concerning the use of information technologies affects the outcome, but focus on participation in business process redefinition. While this is essential, it is not sufficient. We have discussed how goals may be subverted by the technical systems design and implementation processes that follow business process redefinition.

Muller et al. (1993) list a variety of methods for participatory design, classified by the position of the activity in the development cycle and by "who participates with whom in what". The latter axis ranges from "designers participate in users' worlds" to "users directly participate in design activities". For participatory design to *be* participatory, user-worlds must be effectively represented in the design. But, as discussed above, there is a wide disparity in user "worlds". Participatory development has more potential to be politically disruptive and contentious than traditional (non-participatory) forms of system development, because it involves a wide variety of interests, with differing objectives and perspectives on how organizational work and responsibilities should change (Howcroft and Wilson, 2003; Winograd, 1996). This situation is therefore managed carefully in practice. System stakeholders are selected for participation on the basis of political affiliations and compliance, rather than for their understanding of organizational systems support and information requirements. This constrains user choice and significantly affects the potential to achieve a human-centered system design (Howcroft and Wilson, 2003). Users often have little choice about whether to participate. Even when trained in system development methods, users and other non-technical stakeholders often cannot participate on an equal basis with IT professionals (Howcroft and Wilson, 2003; Kirsch and Beath, 1996). User views are often inadequately represented because of cost constraints, or a lack of appreciation of the significance of users' perspectives (Cavaye, 1995). Howcroft and Wilson (2003) argue that the user choice is significantly constrained by organizational managers, who predetermine boundaries for the scope of the new system, who select who will participate in systems development and to what extent.

Because of its reliance on the production of technical system prototypes, the participatory approach is therefore technology-focused. IT professionals frame user perceptions of how a

technology can be employed (Markus and Bjorn-Andersen, 1987). They are able to constrain the choices of non-technical stakeholders, by the ways in which alternatives are presented and implemented in the system prototypes. User worldviews may easily be relegated to "interface" considerations by technical system designers, even when the explicit focus of the method is on joint system definition (Gasson, 1999). The use of participatory design may become a power struggle between, on the one hand, "rational", technical system designers and, on the other hand, "irrational" user-representatives who are unable to articulate system requirements in technical terms (Gasson, 1999; Nelson, 1993). The concept of empowering workers raises hackles: this is seen as "social engineering" that compares unfavorably (in scientific, rationalist discourse) with "software engineering". Designers who engage in such irrational behavior must have a subversive agenda that is counterproductive (Nelson, 1993). Thus, participatory design may often be subsumed to the less intrusive (and much less confrontational) path of producing user-centered design "methods" that can be partially used, in ways chosen and controlled by technical designers.

Interaction Design

Interaction design is a recent development arising from work in Human-Computer Interaction (HCI). It considers a much deeper set of concepts than the traditional HCI interests of user-interface affordance and usability. Interaction design examines the ways in which people will work with a technical artifact and designs the artifact to reflect these specific purposes and uses (Preece et al., 2002). Winograd (1994) defines interaction design as follows:

> " My own perspective is that we need to develop a language of software interaction - a way of framing problems and making distinctions that can orient the designer. … There is an emerging body of concepts and distinctions that can be used to transcend the specifics of any interface and reveal the space of possibilities in which it represents one point." (Winograd, 1994).

So interaction design has the potential to consider a space of possibilities, but in general appears to be constrained to specific interactions with a predetermined technology by the tradition of HCI discourse. Interaction design, as defined by Cooper (1999) -- who claims to have invented the approach -- is "goal-directed design":

> " Interaction designers focus directly on the way users see and interact with software-based products." (Cooper, 1999).

Interaction design from this perspective is product and development driven: this approach defines what software system products should be built and how they should behave in a particular context (Cooper, 1999). But goal-directed approaches are only appropriate when the problem is relatively well-defined (Checkland, 1981; Checkland and Holwell, 1998). Most organizationally-situated design goals are emergent. A similar, goal-driven approach is taken by Preece et al. (2002), who emphasize "the interactive aspects of a product" (*page 11*). Although they extend the goal-driven concept with rich discussions of use, their perspective is also essentially driven by the notion that design is centered around conceptualization of a computer-based product with an *individual* user. Inquiry into the socio-cultural worlds of its use and into negotiated collaboration between interested stakeholders are secondary.

Agile Software Development

Formal methods are increasingly being abandoned in favor of rapid methods with shorter lifecycles and a lower administrative overhead (Barry and Lang, 2003; Beynon-Davies and Holmes, 1998). But rapid methods do not appear to deal well with user requirements and may lead to a more techno-centric focus than with traditional methods (Beynon-Davies and Holmes, 1998). There is a temptation with rapid approaches, for system developers to revert to the code-and-fix approach that characterized software development before the advent of formal methods (Boehm et al., 1984; Fowler, 2003). "Agile" software development was conceived in response to

a perceived need to balance technical system design interests with an understanding of user requirements. Uniquely, this approach is a practitioner-initiated approach to human-centeredness in IS design. Highsmith's (2000) *Adaptive Software Development* and Beck's (1999) eXtreme Programming are both examples of agile software development: practitioner-instigated approaches that combine a minimalist form of system design (i.e. informal methods and short lifecycles) with a user-centered approach. *The Agile Manifesto* (Fowler and Highsmith, 2001) argues for the following points:

- *Individuals and interactions* are valued over processes and tools.
- *Working software* is valued over comprehensive documentation.
- *Customer collaboration* is valued over contract negotiation.
- *Responding to change* is valued over following a plan.

These points reflect many of the conclusions of the literature discussion above, particularly with their focus on goal emergence. The ways in which goals are inquired into, agreed and made explicit are critical to achieving a human-centered outcome. Agile software development emphasizes an adaptive approach to defining system goals and requirements, as the design proceeds. This is an implicit recognition of the difficulties of understanding the needs of multiple user worlds, in advance of the system design. System goals and requirements are adapted to the designer's (and others stakeholders') increasing understanding of the role that the system will play, in organizational work. In *Adaptive Software Development*, Highsmith (Highsmith, 2000) rejects what he terms "monumental software development", in favor of "fitting the process to the ecosystem". At the heart of the approach are three overlapping phases: speculation, collaboration, and learning. He argues that systems design should respond to the contingencies of the local context, rather than fitting the problem analysis to the framework underlying a formal analysis method. Although Highsmith does not prescribe specific methods, he does emphasize teamwork and the involvement of system users in all aspects of system definition and design. However, although Highsmith's work has been influential in forming popular perceptions of how to manage system design, it does not offer a method for performing design. One of the most popular methods for agile software development is *eXtreme Programming* (Beck, 1999). This approach is based partly on the concept of scenario analysis (Carroll and Rosson, 1992) - a concept that is familiar to HCI researchers but novel to many technical system designers. The *eXtreme Programming* approach emphasizes a specific way of eliciting requirements from system users, in an informal and iterative process. Technical systems developers work in pairs with selected users, to generate short scenarios, which are coded into a system prototype. One developer codes, while the other checks the code for authenticity and correctness (these roles are swapped frequently). The user is invited back to validate the prototype against the scenario and to generate additional scenarios, based on their realization of shortcomings or omissions in the original scenario generated, after having used the prototype.

In its focus on emergence and "the people factor", agile software development may be considered human-centered in its intent. However, its ultimate emphasis on the practice and profession of producing software systems, without explicit validation of system goals and organizational roles by non-technical stakeholders, renders it vulnerable to deadline-driven expediency (Nelson, 2002). Agile approaches provide a worthwhile attempt to deal with problems of implicit knowledge, evolutionary learning (by users) of what technology has to offer for their work, and misunderstandings between technical designers and users, as technologists gradually enter the lifeworld of the user. But these approaches are based on the development of software, rather than organizational systems. It involves a very small selection of "representative" users, there is no attempt to understand or investigate the wider, socio-technical system of work and there is little attention paid to the selection of appropriate system users for

scenario generation. Additionally, this method suffers from a common problem of evolutionary prototyping: the approach starts with the specific intention of building a technical system, not with the intention of bringing about organizational and technical change. As Butler and Fitzgerald (2001) remark, stakeholders must be involved in the definition of organizational and process change, before their involvement in IT systems development can be considered anything other than token.

Evolutionary Models of Design

Recent management concern has centered on more human-centered and business-oriented approaches to IS development (Hirschheim & Klein, 1994). However, IS development projects are concerned with process management issues at a macro level, rather than an individual level; an attempt to encompass both macro processes and human and organizational concerns can be seen in the spiral model of software development presented by Boehm (1988), shown in Figure 5. The spiral model is an attempt to manage design emergence, uncertainty and risk in ISD project management. As such, it preempts many of the situated action issues discussed above. In this model, the radial dimension represents the cumulative cost of development to date, the angular dimension represents the progress made in completing each cycle of the spiral.

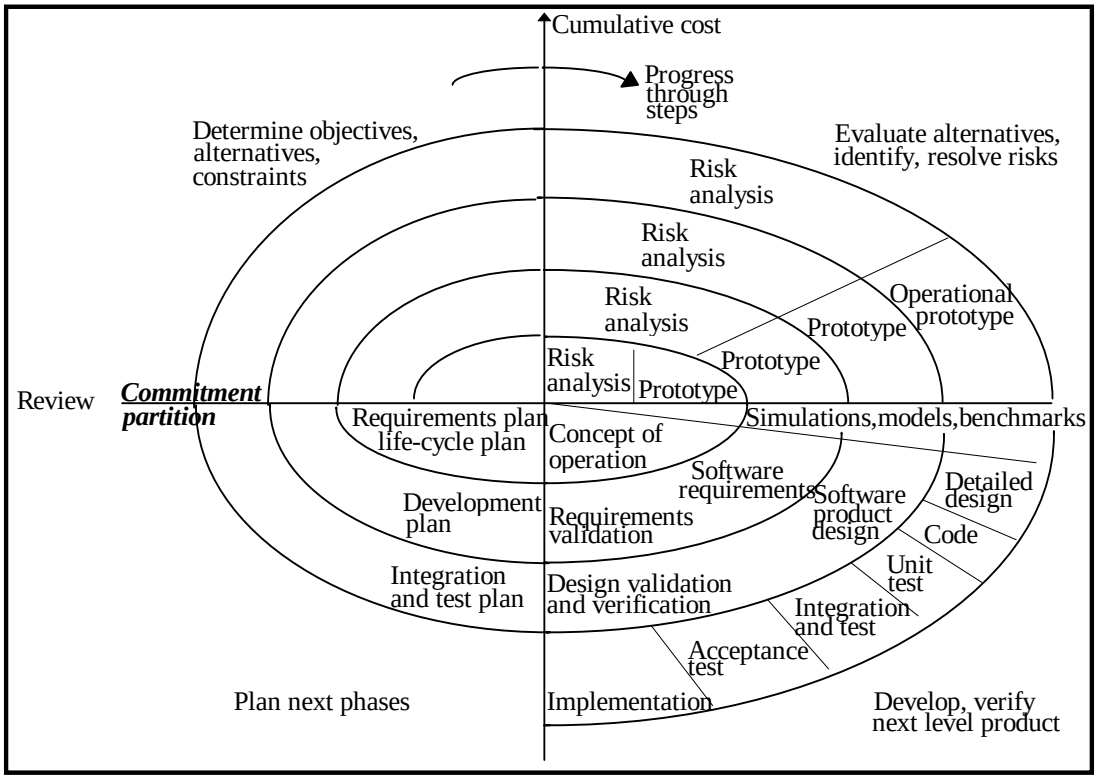

Figure 5: The Spiral Model Of Software Development
(Source: Boehm, 1988)

An underlying concept of this model is that each cycle involves a progression that addresses the same sequence of steps, for each portion of the product and for each of its levels of elaboration. However, there are three main problems with this model in guiding the management of IS development:

1. The model represents a macro level representation of development: it does not address outputs from effective design processes and it does not represent the behavioral issues which managers face in real-life IS development

2. The skills required at different stages of the cycle are unrealistic: it is not feasible to expect the same group of developers to possess (or to acquire) skills in both detailed technical design and risk analysis
3. The model cannot be said to represent IS development practice, even at a macro level: Boehm (1988) admits that it is not based on empirical observations, nor has it been tested experimentally.

Despite these criticisms, the model is a real advancement in theoretical thinking about IS development practice. It embodies an iterative process and encompasses human and organizational concerns through the inclusion of evolutionary prototyping as an essential component of organizational risk management. However, the four evolutionary stages of the model - determine objectives, evaluate alternatives, develop product and plan next phase - may be too akin to the "rational" model of decision-making (Simon, 1960), criticized in the previous section, to be of help in managing real-life processes. What is needed are models which encompass both macro business processes and human and organizational concerns but rely less on managing the predictability and rationality of process outcomes. There is a basic conflict here: professional management is concerned primarily with the reduction of risk through an emphasis on predictability and an assertion of rationality, while effective design requires the "control and combination of rational and imaginative thought" (Lawson, 1990). It might be that the two goals are incompatible: that different models are required for the control of macro (project) processes and the support of micro (design) processes. However, the two are closely interlinked and any approach to IS development which adopts a single perspective will not succeed.

The need for both deductive and inductive reasoning in design has to do with how human beings cope with design requirements that are either undiscovered or tacit in nature. So it would follow that the role of tacit knowledge and inductive reasoning in design is greater for problems that are not well-structured. Schön (1983) describes design as "art". A design problem can only be approached via "reflection-in-action": purposeful action which calls on tacit knowledge for its execution. The concept is best described in Schön's (1983) own words:

> " Even when he [the professional practitioner] makes conscious use of research-based theories and techniques, he is dependent on tacit recognitions, judgments and skillful performances."
> *(Schön, 1983, page 50).*

Expertise in professional skills such as design can only be accomplished be learning-through-doing (Schön, 1983). The role of learning-through-doing was also highlighted by Rosenbrock (1988), in discussing the exigencies of engineering design and by Argyris (1987), who called for a new way of approaching information systems design. But if we see learning as central to design, the hierarchical decomposition approach becomes unusable. Empirical studies observe "opportunistic" design strategies (Jeffries et al., 1981; Guindon, 1990a, 1990b; Khushalani et al., 1994), which is defined as (various types of) deviation from hierarchical decomposition.

Visser and Hoc (1990) argue that many of the early studies into design processes conflate prescription and description: they ignore what the activity of design is really like, to focus on what it *should be* like. In addition, early studies often presented subjects with a unitary, relatively well-defined and well-structured problem to solve. Hierarchical decomposition is excellently-suited to this type of problem and "opportunistic" deviations from a decompositional strategy must be considered unproductive. But when designers are faced with ill-bounded and ill-structured problems, decompositional strategies fail.

3.3 Empirical Studies Of Human-Centered and Evolutionary Design Processes

The "user-centered" model of evolutionary prototyping. (ref. Gasson's debunking of this model).

organizational context.

4. DESIGN DOMINATED BY BUSINESS PROCESS CONSTRAINTS

Section 4 presents the extension of design theory to encompass situated, collaborative, group action and discusses this extension as a response to strategic business coordination constraints. Business processes are viewed as cross-functional collaboration between different work-groups, whether internal to, or external to the organization. This fourth perspective moves the design of information systems on from the limited perspectives offered by viewing an information system as synonymous with a computer system and resolves many of the theoretical conceptualization issues implicit in recent IS design writings.

5. Theories of Group Action In Design

Section 5 deals with the extension of design theory to collaborative, group action.

Weick (1979) argues that shared cognition emerges through the process through which a group develops collectively structured behavior and that this process is inconsistent with achieving intersubjectivity. He describes four phases of this process, shown in Figure 6.

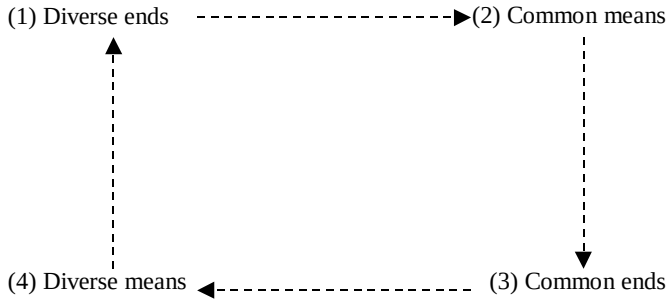

Figure 6: Weick's (1979) model of group development

Initially, groups form among people who are pursuing diverse ends. As a structure begins to form, group members reciprocate behavior which is valued by other group members while still pursuing individual goals and thus converge on common means: common group process rather than common goals. Once the group members converge on interlocked behavior, a shift occurs from diverse ends to common ends. Initially, the common end is to perpetuate the group's collective structure, which has been instrumental in aiding individuals to get what they want; other common ends follow from this recognition of mutual interest. Finally the group is enabled to pursue diverse means, often because of division of labor between group members permits individuals to pursue ends in ways which fit with their own specialization, but also because the stability engendered by a durable collective structure enables individuals to pursue elements of the problem-situation which appear unpredictable and disorderly in comparison to the stable environment produced by the group. There may be pressures to reassert individuality following the subsumation of individuals' interests to those of the group.

Distributed Cognition In Design

An explanation of how the division of labor identified by Weick may be enabled lies in the notion of distributed cognition (Norman, 1991; Hutchins, 1991). Distributed cognition involves a model of the task or problem in hand which is "stretched over" rather than shared between members of a collaborative group (Star, 1989). Distributed cognition enables members of design teams and other workgroups to coordination their actions without having to understand every facet of the work of other individuals in the group.

> " Distributed cognition is the process whereby individuals who act autonomously within a decision domain make interpretations of their situation and exchange them with others with whom they have interdependencies so that each may act with an understanding of their own situation and that of others." *(Boland et al., 1994, page 457)*.

The acquisition of knowledge by design teams involves both *shared* cognition and *distributed* cognition. The concept of shared cognition is illustrated in Figure 7: this diagram illustrates the extent of intersubjectivity (shared meanings) between organizational actors.

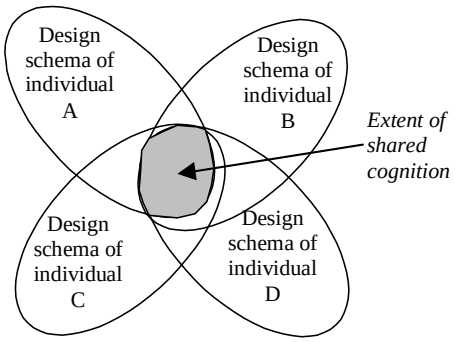

Figure 7: The Concept Of Shared Cognition (adapted from Laukkanen, 1994)

Design depends upon intersubjectivity for effective communication between team members to take place (Flor and Hutchins, 1991; Hutchins, 1990, 1991, 1995; Orlikowski & Gash, 1994; Star, 1989). Technical system designers, "successful in sharing plans and goals, create an environment in which efficient communication can occur" (Flor and Hutchins, 1991). Orlikowski & Gash (1994), in a hermeneutic analysis of different interest groups' assumptions, knowledge and expectations of a new groupware technology, refer to intersubjectively-held mental models as "shared technological frames":

> " Because technologies are social artefacts, their material form and function will embody their sponsors' and developers' objectives, values, interest and knowledge regarding that technology" *(Orlikowski & Gash, 1994, page 179).*

Distributed cognition may be coordinated using "boundary objects" which represent the current state of the outcome of group activity (Norman 1991; Star, 1989). Boundary objects which aid distributed cognition include external representations of a design (e.g. a diagrammatic model) and design specifications. Thus, intersubjective understandings of the design problem and solution are not required. An effective collaborative group can function well when group members share very little common understanding of the problem in hand, if they have effective coordination mechanisms (Star, 1989).

Individual group members may have very different models of the organizational "world" and different design goals. The literature on collaborative group work normally assumes intersubjectivity - a common vision shared by all group members. Distributed cognition theorists would argue that a group of designers do not need to understand all the elements of the design problem. They just need to achieve sufficient overlap between their different perspectives and understandings of the design problem for the group to coordinate their design activity. Although some writers have proposed that group coordination may be aided by the use of "boundary objects" (Hutchins, 1991; Norman 1991; Star, 1989) we know little of the mechanisms by which effective distributed cognition is achieved and maintained. Lave (1991) argues that the process of socially shared cognition should not be seen as ending in the internalization of knowledge by individuals, but as a process of becoming a member of a "community of sustained practice". Such communities reflect the sociocultural practices of the group in its

Design As Situated Creativity

The Role of Tacit Knowledge About The Application Domain

Argyris and Schön (1978) had previously compared the espoused theory held by a person to explain how they performed a task, and the theory-in-use, what they actually did to perform the task. Espoused theories tended to conform to explicit organizational procedures and rules, while

theories-in-use tended to be derived from implicit understanding of a task's requirements, which were difficult or impossible to articulate. Schön's (1983) work developed this concept with a focus on design (among other work-activities). He argued that design depended upon continual interaction with the problem-context, followed by reflection upon that interaction. Learning through doing is key to this perspective.

The Role Of Social Construction In Design

A further thread in the individual perspective of design is provided by Mackenzie and Wajcman (1985), Bijker et al. (1987) and Latour (1987, 1991), building on the social construction theories of Berger and Luckman (1966). These authors argue that technology is socially constructed and that features that enforced a particular behavior or control mechanism were embedded in the form of technology designed for a specific context. Mackenzie and Wajcman (1985) argue that this mechanism is unconscious: the form of new technology is constrained by the form of technological exemplars encountered by the designer. But Latour (1987) argues that the design of technology embeds the explicit intentions of the stakeholders whose interests the designer serves. Scarbrough and Corbett (1991) conclude that, while technology does serve the interests of dominant stakeholders, this is because of a cyclical influence. Technology is shaped by social constructions inherent in the context of design, as shown in Figure 8.

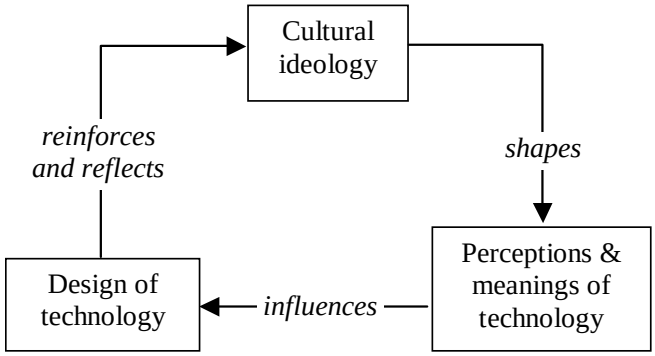

Figure 8: The cycle of design influences (*adapted from* Scarbrough and Corbett, 1987)

This model of design influences is significant because of the dependence that it demonstrates between the cultural context and the design of technology. Taken with Latour's (1987) work on actor-network theory, which demonstrated a network of causality between the dominant interests and technology, this model provides a convincing argument for how the cultural management of meaning (Smircich and Morgan, 1982). Perceptions of the meaning of technology to the organization (for example, its role and value) influence the design of technical systems, which in turn reinforces the cultural ideology of the organization, which in turn shapes and manages meanings of technology within the organization … and so on. Design is thus both formed by and forms the social context in which it takes place.

Design As Situated Action

Suchman (1987) likened design and planning to steering a boat: while the overall goal may be fixed, the path to achieve that goal is affected by the local contingencies (the waves and currents that are encountered on the way to the goal). Current design practice is constrained by a view of information systems as rule-based information-processing systems, where human work disappears from view. She argues that design can only succeed if the process permits goals to change and contingent processes to emerge. This is reminiscent of Mintzberg's (1985) arguments concerning strategic planning. As any plan is executed, new contingencies arise that cause some parts of the previous strategy to be discarded and new components to be added. This often leads to a change in the detailed goals of the plan. The consequence of applying a situated action model of human problem-solving to design is shown in Figure 9.

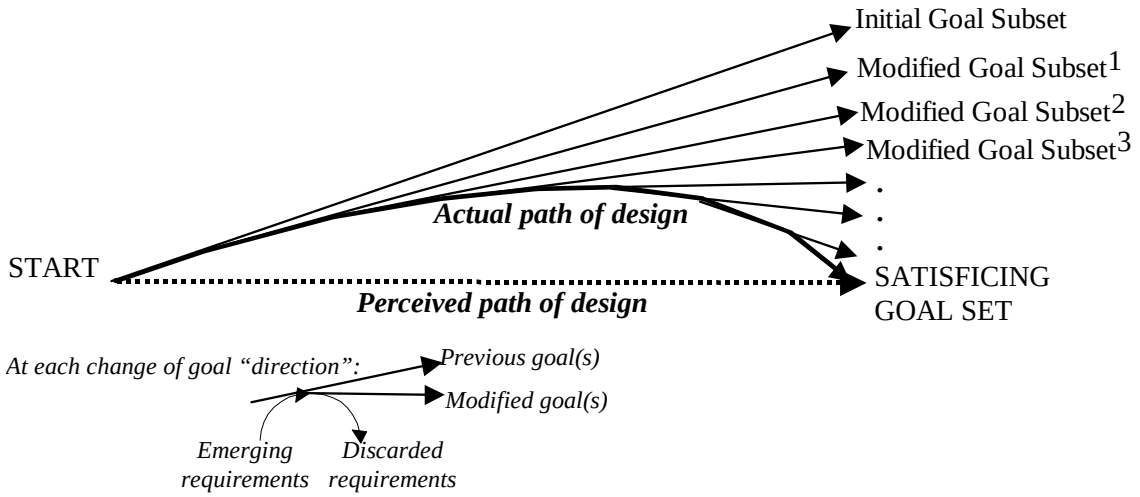

Figure 9: The Implications Of Situated Design

Design is this no longer "guided" by goals, but a relatively unpredictable process of seeking out short-term and partial goals. As postulated in Gasson (1998), design is conducted by taking "good enough", partial goal subsets and working with these until situational contingencies or a conflict of explicitly stated requirements with an individual's implicit model of the way that the "real world" works causes a redefinition, in part of the design goals. The latter type of conflict, referred to by Winograd and Flores (1982, *after* Heidegger, 1960) as a cognitive *breakdowns*, underlies a groundbreaking study of design by Malhotra et al., discussed in the empirical section below.

The emergent, situated model of design is a significant development in how design is perceived. The majority of extant design theories are goal-oriented – including many of the more recent "soft" approaches. For example, Checkland's 1981 Soft Systems Methodology is based on the notion that stakeholders in the new information system being designed are capable of defining consensus outcomes that they wish the new system to achieve. But consensus may mean that perspectives are unitary in nature, reproducing a primary constraint of traditional approaches to IS design. Burrell (1983) criticizes the Soft Systems Approach of Checkland (1981) for privileging the management interest through the modeling of consensus, which is unrealistic in a political context where management interests dominate. In contrast, goals constantly evolve with an understanding of the design and the actual path of design is much more complex (and longer) than that perceived by actors external to the design process, who only see the start and end points of the design. This model may explain why timescales always 'slip' in IS development projects - a common comment from those not involved in such projects is "why did it take you so long?". A critical process of design must therefore be the management of external perceptions of the design process, particularly those of the "global network" (Law & Callon, 1992) - the informal network of influential decision-makers affected by, and indirectly attached to a design project.

Socially Situated Action In Communities of Practice

An important development of situated action theories arose from arguments that individual action is situated within one or more communities of practice (Brown and Duguid, 1991; Lave 1991; Lave and Wenger, 1991). Brown and Duguid (1991) argue that learning takes place within a sociocultural context: a set of rules, norms and expectations that are constructed by members of the local work community, through their interactions. Lave and Wenger (1991) demonstrate that tasks and artifacts cannot be abstracted independently of their sociocultural context. For example, Brazilian street children who cannot perform mathematical calculations in a classroom context are perfectly capable of performing the same calculations when transacting a sale. Understanding is situated in the context of practice. Remove the understanding of the context in

which the task will be performed and you remove the ability to understand and abstract the task. Lave and Wenger also argue that membership of a community of practice depends upon the implicit learning and adoption of the sociocultural norms of that group. This results in a great deal of organizational practice which is not rational, but historical in nature. Information system designers need to understand the sociocultural practices underlying the day to day practice of work-tasks, to be able to design effective support for those practices.

Organizationally-Situated Processes

Political and Organization-Related Models

The core role played by information system designers in mediating social and political concerns, and their unpreparedness for this task, was demonstrated by Boland and Day (1989). An IS designer was shadowed and then interviewed about her work. She expressed her concern at having to make decisions about social and political issues which she saw as outside an appropriate scope of work for IS design. Such issues were often dealt with at an implicit level: the designer was not aware of making such decisions until much later.

Gasson's (1998) organizational context model?

6. SUMMARY AND DISCUSSION

Section 6 summarizes the evolution of design theories, discusses some lacunae in current understandings of how design works and presents a dual-cycle model of design, based on an empirical field study of situated design, to resolve some of the major deficiencies in current information system design theory.

6.1 The Adoption Of Successive Theories of Design

As might be expected, the information system design literature follows the paradigmatic assumptions of the period during which they were conducted. Early studies assume an individual, rational, "information processing" view of human cognition (Mayer, 1989) and focus on the design process as unitary, structured problem-solving. Studies in the 1970s and early 1980s follow Simon (1973) in their focus on structuring a unitary problem (e.g. Gane and Sarsen, 1979). Most studies equate the technical information system with an organizational information system and so are surveys of what IT system development methodologies [sic] are in use. Even after ground-breaking theory in socio-technical systems (Emery and Trist, 1960) had been applied to information system design by Enid Mumford (Mumford, 1983; Mumford and Sackman, 1985), empirical studies continued to focus uncritically on applying structured IT system development methods (e.g. Sumner and Sitek, 1986; Necco, 1989).

This emphasis is followed by the early "psychology of programming" interest group literature (e.g. Hoc et al., 1980). Early empirical studies in this area found a distinct difference between the problem decomposition strategies employed by novice vs. experienced programmers (Adelson and Soloway, 1985; Jeffries et al., 1981; Kant and Newell, 1984). Novice programmers tended to employ a top-down, depth-first decomposition strategy (i.e. hierarchical decomposition, as shown in Figure 1 above). Experienced programmers employed a top-down, breadth-first strategy, developing an integrated approach to the synthesis of solutions to partial problem-statements in an ad hoc manner. They conclude that (experienced) design is therefore *opportunistic* in nature. Most of these studies employed a very structured design problem, such as designing a program to control the priorities of an elevator. They also often used students as proxy subjects for "novice" and "experienced" programmers, which limits the validity of their findings.

From this quote, it can be seen that an evolution is perspective has taken place: an “information system” is now viewed as a work-system. Support for the IT system user’s decision-processes

and a focus on the user-interface with the IT system characterize this period of IS design literature. This period contains a great many studies that focus on the application of prescriptive methods for ensuring usable systems, in terms of task-fit and IT system interface design. "Human-centeredness" and emancipation become popular threads in the IS and IR-related organizational change literature (*for example:* Corbett, 1987; Gill, 1991; Scarbrough & Corbett, 1991; Zuboff, 1988) . Prototyping becomes a popular method for developing IT systems (e.g. Boehm, 1984; Floyd, 1984, 1987). But prototyping does not necessarily focus on work-task support. Floyd (1984) contrasts evolutionary prototyping, an approach that involves IT system users in evaluating and providing input to the design of an evolving system, with experimental prototyping, the production of prototypes for feasibility testing of a technical concept. But the perspective of design is seriously constrained by a focus on the "user" of a technical system, rather than on the combined social, work-process, business strategy and technical goals of an organizational information system.

Social construction is brought into the explicit processes of design by the "soft" systems perspective (Checkland, 1981). Churchman's (1971) work on inquiring systems drew attention to the interconnectedness of system elements and requirements, and the need for purposive inquiry into the problem situation to define design goals. Checkland (1981, Checkland & Scholes, 1990) built on this concept to provide an holistic theory of information system design that has four main properties:

1. An organizational information system is a system of *human-activity* supported by an IT system. The task of information system design is to investigate the problem-situation concerning a particular human-activity system and to determine appropriate interventions, only some of which may involve IT system design.
2. Any human-activity system involves multiple sub-systems of tasks, performed for multiple purposes. A major priority for information system design is for the designer, participants and other stakeholders in the human-activity system to understand and to separate conceptually the *purposive* systems that constitute the whole.
3. Different information system stakeholders have different worldviews that cause them to interpret the meaning and purpose of human-activity in different ways. To succeed in implementing an information system that will benefit the majority of stakeholders, the design process must focus on the collective negotiation of requirements for action.
4. Selection of appropriate scope(s) for organizational intervention(s), such as the design of a new information system (work-task changes plus IT system support), should be made explicit to all system stakeholders and subject to consensus agreement.

Criticisms may be leveled at the detailed mechanisms proposed by Checkland and other soft systems authors: for example, Burrell (1983) criticized Checkland's (1981) notion of consensus, arguing that the facilitated workshops proposed for this purpose would always privilege the management interest over other interests). But Checkland's work has had a significant influence on both theories and practice and is responsible for changing the dominant paradigm of information system design. The negotiation and incorporation of a consensus set of system requirements, derived from multiple stakeholder worldviews is a very long way from Alexander's (1964) argument that the structure of a system solution is embedded in the system problem and exists independently of the designer. Checkland's work has given a deeper meaning to Rittel's (1972) argument that design is cognitively-constructed and so should be derived through "argumentation".

This perspective can be contrasted with the hard systems approach, which sees system properties as being objective, rather than emergent, with communication and control being human interactions with the material (computer-based) 'system', rather than properties of the system itself. While soft systems approaches to IS design see IT as the "serving system" to a "served

system" of purposeful human-activity (Winter, Brown and Checkland, 1995), hard systems approaches see IT as the target object system. However, the view is still static: the soft systems literature views design as being a process of negotiating a consensus on organizational system definitions, which embody structure and persistence. It may also be argued that the whole thrust of the 'problem' investigation literature in the field of IS is aimed at structuring problems and constructing structured data (Preston, 1991). An alternative model rejects organizational structure as the basis for design (Truex & Klein, 1991): organizations are seem as emergent and dynamic, with design defined as situated, evolutionary learning.

More recently, information system design is viewed as socially-embedded. The work of Winograd and Flores (1986) argued that design includes "the generation of new possibilities" in an organizational change context. They provided significant insights into the nature of cognition in design and its social context. Brown and Duguid (1990) take this perspective further with a discussion of design as supporting communities of practice. They view design as socially-situated and emergent: for successful design, the IS designer should be a peripheral participant in the community of practice which the information system is intended to support.

6.2 Limitations Of Current Design Practice Arising From The Various Literatures On Design

It would appear, from the review presented here, that design theories are primarily concerned with problem closure. While this may have been appropriate at a time when information technology designers were concerned with relatively well-defined, unitary problems, it is no longer appropriate for groups of designers engaged in the exploration and definition, as well as the solution of, "wicked" problems relating to organizational information systems. The problem of "the problem" dominates design theories and yet design models are concerned more with solution definition than with problem investigation. Given the concerns expressed above, coupled with the limitations of human cognition, it would appear that evolutionary models of the design process are more appropriate for

Thus we end with five areas of concern, that limit current conceptualizations of design. As with any "wicked" problem, these five areas may be conceptually separated, yet are interrelated.

1. *The labor process problem:*

   While the traditional model provides a clear basis for managing the labor process in IS development, it artificially separates the conceptual and social processes of organizational IS development which are referred to here as design processes. Design activity cannot be separated into a single stage of the system development lifecycle, as in the traditional model: requirements specification, design and technical system implementation are intertwined (Bansler and Bødker, 1993) and so require support and legitimacy at all stages of the system development life-cycle. Radical redesign of a technical system may occur even at the system implementation stage, when problems are encountered during interactive user testing; such redesign is often referred to euphemistically as 'system maintenance' (Lientz and Swanson, 1980).

2. *The design process-model problem:*

   The way in which design is managed is based upon a decompositional, breadth-first exploration of the design problem, where all requirements for a solution are defined before problem decomposition is attempted. But empirical studies of individual design strategy show that design strategies are "opportunistic" in nature, adopting depth-first, iterative, recursive or 'inside-out' approaches (Ball and Ormerod, 1995). Turner (1987) argues that "requirements and solutions migrate together towards convergence". Designers fit known solutions to parts of the problem, or reframe the problem to fit known solutions (Malhotra et al., 1980; Guindon, 1990; Turner, 1987).

3. *The bounding problem:*

   The traditional model presupposes a design problem which is unitary in nature, which exists independently of the designer's frame of reference and which is capable of analysis under conditions of "bounded rationality" (Simon, 1973), where the designer bounds the problem until it is amenable for structured analysis. But the design of complex organizational information systems centers upon the investigation of socially-constructed, "wicked" problems (Rittel and Webber, 1973), which are associated with interrelated, organizational systems of activity. Such problems cannot be "stated" or "solved" in the sense of definitive rules or requirements for a solution (Moran and Carroll, 1996): they are socially-constructed and subjective (Schön, 1983; Galliers and Swan, 1997) and each problem is interrelated with – and thus cannot be defined separately from – multiple, other organizational problems (Rittel and Webber, 1973).

4. *The collaboration problem:*

   The traditional model of IS design is based upon an individual, cognitive model of problem-solving and so excludes many necessary social processes required for collective investigation and negotiation of design attributes. Empirical studies indicate the centrality of communication, shared learning and project co-ordination, but such processes are often deemed illegitimate by managers guided by traditional, individual models of design (Curtis et al., 1988; Walz et al., 1993). Existing approaches resolve this problem by assuming that a unitary, intersubjective model of the designed system can be negotiated by design team members. As we have argued above, this may not be feasible in most organizational information systems supporting complex human work-systems, as these problems are "wicked" problems (Rittel, 1972), that are socially constituted, represent multiple work-goals and are highly interrelated.

5. *The context problem:*

   The traditional model ignores the context of design, as situated in a socially-constituted organizational culture. The form taken by a design involves both technical and social issues, for example, designers may debate the form of a technical artifact in terms of whether users should be prevented by its design from amateur repairs, or whether its design should reflect users' desires for conspicuous consumption (Callon, 1991). Design is also political: an information system may change the nature of work and the basis of power, for different stakeholder groups within an organization (Wilkinson, 1983). Design processes are viewed as irretrievably interrelated with context: design activity is "situated" in knowledge and assumptions about organizational contexts (Gasser, 1986; Suchman, 1987; Lave, 1991; Lave and Wenger, 1991). Legitimate system "solutions" to political, situated problems are negotiated, rather than defined and are emergent, rather than explicitly stated (Boland and Tenkasi, 1995).

The five problems of design result in a separation of degree, rather than concept, between the design of a physical artifact and the design of an information system. Current models of design focus on design closure and so delegitimize the essential activities of investigating, negotiating and formulating design problems. We need to focus on "opening up" the design problem, to legitimize the modes inquiry required for effective design of complex, situated information systems. An understanding of this dialectic has significant implications for both the research and practices of design. The situated nature of design requires design models to be constructed through sharing simulated design contexts, rather than through the medium of abstract representational models; this is ill-supported by traditional methods for design. Such constructions cannot be shared intersubjectively, but rather are distributed between collaborating design-group members. Additionally, the contextual constraints upon IS design are considered to have significant implications for design and constitute a critical area of activity which should be

managed proactively, particularly where influential organizational decision-makers are involved as stakeholders in a design initiative. These findings have implications for co-operative learning, knowledge management and organizational innovation. If organizational problem-investigation processes are seen as involving distributed knowledge, then the focus of organizational learning and innovation shifts from *sharing intersubjective organizational knowledge* (achieving a "common vision") to *collaborating in constructing distributed organizational knowledge* which is emergent, political and incomplete.

6.3 A Dual-Cycle Model of Situated, Information System Design

Section 8 concludes the discussion by presenting a dual-cycle model of design, based on this review of current knowledge, to resolve some of the major deficiencies in current information system design theory.

Resolution Of The Five Problems

This can only be highlighted the central role played by a periodic reopening of the design problem-definition in achieving shared understanding.

Findings From Previous Studies:

1. Effective, shared design understanding of design needs results from *repeated* revisiting of design problem definitions by stakeholders. **This is driven through the introduction of a new 'primary generator' idea.** Too early a closure of the problem is counter-productive.
2. It is not feasible for each member of a collaborative design team to understand all of the design rationale for a complex organizational information system. Readiness for problem closure (and solution specification) is gauged by the extent to which the team share an understanding of organizational goals and outcomes, not by the extent to which they share an understanding of the designed system.
3. The effectiveness of a design is constrained by the need to manage external perceptions and expectations of design outcomes. Successful expectation-management is key to successful evolution of the design, as stakeholder understanding of the design problem improves.

The extent to which a shared understanding of organizational goals was found to be more critical than a shared understanding of design outcomes to group perceptions of design completeness: an initial, dual-cycle model of collaborative design processes was proposed (Gasson, 1998a). The model is shown in Figure 10.

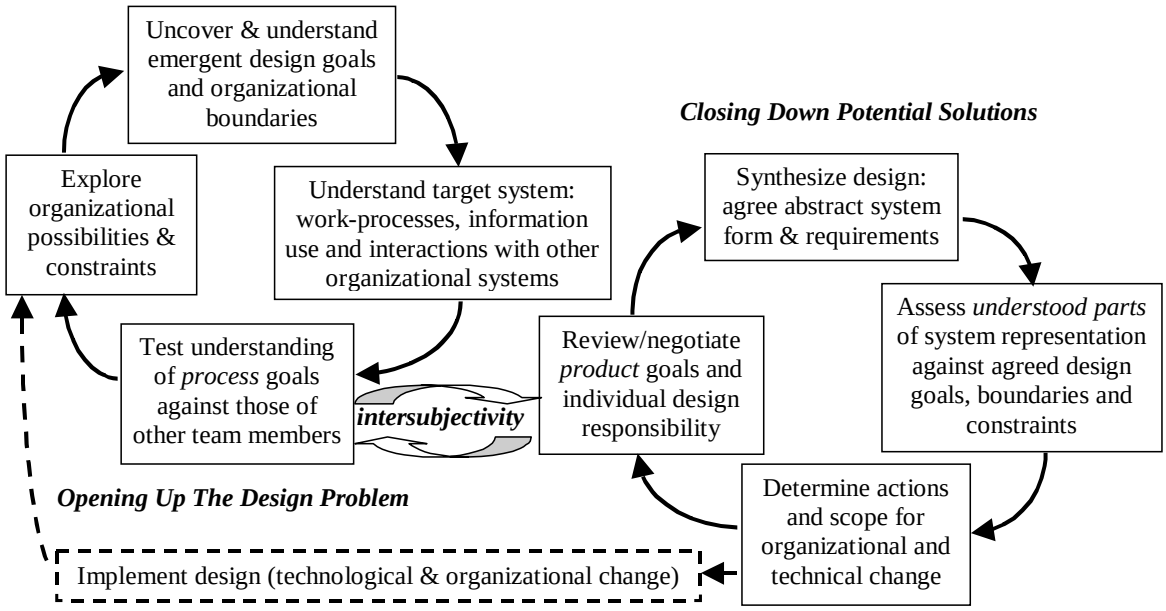

Figure 10: A Dual-Cycle Model of Collaborative, multi-domain Information System Design

Show progress and iterations of the following process model of design:

- opening up activity: requires detailed investigation of how the organization works (not just how individual domains work) - need to understand target system in terms of work-processes, information use and interactions with other organizational systems of work - in doing so, team members construct process goals (how they want to affect the organization), *which enable them to effectively move into second loop of operation.*
- progress (delegation) only possible once team-members trust each other enough to hand off parts of design - happens only when process goals are shared - then move on to closing down loop.
- breakdown (collective) occurs when product goals conflict with what has been designed - need to reframe process goals collectively in terms of expanded understanding of role of designed process in larger organizational process - this requires return to original loop of operation (opening up), to define new goals and boundaries.
- once new process goals are agreed, can move back to closing down loop, to division of labor in designing details of new process and determining collective action to implement this.

7. IMPLICATIONS FOR RESEARCH AND PRACTICE

The dialectic expressed by the dual-cycle model is critical in our understanding of what design is. It is counterproductive to artificially separate problem articulation from solution formulation. *Problems and solutions converge*, at an individual level, at a group level and in the politically-negotiated organizational processes that surround design activity. Current models of design focus on design closure and so delegitimize the essential activities of investigating, negotiating and formulating design problems. A dual-cycle model of design is proposed: one that focuses on "opening up" the design problem, as much as design closure. An understanding of this dialectic has significant implications for both research and practice of design that are discussed at the end of the paper. The implications for research are that this type of model needs to be investigated in practice and its contingencies understood. How well does this type of “dual-cycle” model fit with the activities required for effective design? What elements drive this type of model, how do we ensure an effective cycling between inquiry and closure, and how do we recognize design stopping points? The implications for practice are that we need to understand the contingencies of this type of model for IS design process management. How do we assess progress, for such an iterative model? How do we plan IS design and development projects, in a way that ensures agreement from project sponsors and a definition of interim deliverables? These questions remain to be answered.

References

Ackoff, R.L. (1974) Redesigning The Future, Wiley

Adelson, B. and Soloway, E. (1985) 'The role of domain experience in software design', IEEE Transactions In Software Engineering, Vol. 11, pp 1351-1360

Alexander, C. (1966), 'A City Is Not A Tree, Design, No. 206, February 1966, pp.46-55

Alexander, C. (1999) 'The origins of pattern theory: The future of the theory, and the generation of a living world', IEEE Software Sept-Oct. 1999, pp 71-82.

Argyris, C. & Schön, D. (1978) Organizational Learning: A Theory Of Action Perspective, Addison-Wesley, Reading, Mass.

Argyris, C. (1987) ‘Inner Contradictions In Management Information Systems’, in R. Galliers, Information Analysis, Selected Readings, Addison-Wesley

Ball, L.J. & Ormerod, T.C. (1995) Structured and opportunistic processing in design: a critical discussion’, Int. Journal of Human-Computer Interaction, Vol. 43, pp 131-151

Bansler, J.P. & Bødker, K. (1993) ‘A Reappraisal of structured analysis: design in an organizational context’, ACM Transactions on Information Systems, Vol. 11, No. 2, pp. 165-193

Barry, C. and Lang, M. (2003) "A comparison of 'traditional' and multimedia information systems development practices," Information and Software Technology (45:4), 217-227.

Beck, K. (1999) eXtreme Programming Explained: Embrace Change, Reading, MA: Addison-Wesley.

Berger, P.L. and Luckman, T. (1966) The Social Construction Of Reality: A Treatise In The Sociology of Knowledge, Doubleday & Company Inc., Garden City, N.Y.

Bertanlaffy, L. von (1968) General System Theory, Braziller, UK

Beynon-Davies, P. and Holmes, S. (1998) "Integrating rapid application development and participatory design," Software, IEE Proceedings (145:4), 105-112.

Bijker, W.E., Hughes, T.P. and Pinch, T.J. (Eds.) (1987) The Social Construction of Technological Systems, New Directions in the Sociology and History of Technology, MIT Press, Cambridge, MA

Boehm, B.W., Gray, T.E. & Seewalt, T. (1984) 'Prototyping Versus Specifying: A Multiproject Experiment', IEEE Transactions on Software Engineering, Vol. SE-10, Number 3, pp 290-302

Boehm, B.W., Gray, T.E., and Seewalt, T. (1984) "Prototyping Versus Specifying: A Multiproject Experiment," IEEE Transactions on Software Engineering (SE-10:3), 290-302.

Boehm, B.W. (1988) 'A Spiral Model Of Software Development And Enhancement', IEEE Computer Journal, May 1988

Boland, R. and Day, W.F. (1989), The experience of systems design: a hermeneutic of organisational action, Scandinavian Journal of Management, 5,2 87-104

Boland, R J, Tenkasi, R. V., Te'eni, D. (1994) Designing Information Technology to Support Distributed Cognition, Organization Science, Vol 5, No 3, pp 456-475

Boland, R J and Tenkasi, R V (1995) Perspective Making and Perspective Taking in Communities of Knowing, Organization Science, Vol 6, No 4, pp 350-372

Brown, J.S. & Duguid, P. (1991) "Organizational Learning and Communities of Practice: Toward a Unified View of Working, Learning, and Innovation", Organization Science, Vol.2, No. 1, pp. 40-57.

Burrell, G. (1983) 'Systems Thinking, Systems Practice: A Review', Journal Of Applied Systems Analysis, 10.

Butler, T. and Fitzgerald, B. (2001) "The relationship between user participation and the management of change surrounding the development of information systems: A European perspective," Journal of End User Computing (13:1), 12-25.

Callon, M. (1987) 'Society in the making: The study of technology as a tool for sociological analysis', in W.E. Bijker, T.P. Hughes, and T.J. Pinch (Eds.) The Social Construction of Technological Systems, New Directions in the Sociology and History of Technology, MIT Press, Cambridge, MA

Callon, M. (1991) 'Techno-Economic Networks and Irreversibility', in J. Law (Ed.) A Sociology of Monsters. Essays on Power, Technology and Domination, Routledge, London, UK

Carroll, J.M. and Rosson, M.B. (1992) "Getting Around The Task-Artifact Cycle How To Make Claims and Design By Scenario," ACM Transactions on Information Systems (10:2), 181-212.

Cavaye, A.L.M. (1995) "User Participation In System Development Revisited," Information & Management (28:5), 311-323.

Checkland, P. (1981) Systems Thinking, Systems Practice, John Wiley & Sons, Chichester.

Checkland, P. and Holwell, S. (1998) Information, Systems and Information Systems: Making Sense of the Field, Chichester UK: John Wiley & Sons.

Checkland, P. & Scholes, J.(1990) Soft Systems Methodology In Action, John Wiley & Sons, Chichester

Churchman, C. W. 1971. The Design of Inquiring Systems: Basic Concepts of Systems and Organization. Basic Books , New York, NY

Cohen, M.D., J.G. March and J.P. Olsen (1972) "A Garbage-Can Model of Organizational Choice", Administrative Science Quarterly, Vol. 17, 1-25

Cooper, A. (1999) The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How To Restore The SanitySams Publishing.

Corbett, J.M., Rasmussen, L.B. & Rauner, F. (1991) Crossing the Border: The social and engineering design of computer integrated manufacturing systems, Springer-Verlag, London

Darke, J. (1979) 'The Primary Generator And The Design Process', Design Studies Vol. 1, No 1. Reprinted in N. Cross (Ed.) Developments In Design Methodology (1984), J. Wiley & Sons, Chichester, pp 175-188

Emery, F.E. & Trist, E.L. (1960) 'Socio-Technical Systems' in C.W. Churchman & M. Verhulst (Eds.) Management Science Models and Techniques, Vol. 2, Pergamon Press, Oxford, UK

Floyd, C. (1984) 'A Systematic Look At Prototyping', in R.Budde, K.Kuhlenkamp, L.Mathiassen, & H.Zullighoven (Eds.) Approaches To Prototyping, Springer-Verlag Books

Floyd, C. (1987) 'Outline of a Paradigm Change in Software Engineering' in G. Bjerknes, P. Ehn & M. Kyng (Eds.) Computers and Democracy: A Scandinavian Challenge, Avebury Gower Publishing Company, Aldershot, UK

Fowler, M. (2003) "The New Methodology," Online at http://www.martinfowler.com).

Fowler, M. and Highsmith, J. (2001) "The Agile Manifesto," Software Development Magazine).

Friedman, A.L. (1990) ' Four Phases of Information Technology - Implications for Forecasting IT Work', Futures, Guildford, Vol. 22, No 8., pp. 787-800.

Gane, C. & Sarsen, T (1979) Structured Systems Analysis: Tools and Techniques, Prentice-Hall, New Jersey

Gasson, S. (1998) 'Framing Design: A Social process View of Information System Development', in Proceedings of ICIS '98, Helsinki, Finland, December 1998, pp. 224 - 236.

Gasson, S. (1999) "The Reality of User-Centered Design," Journal of End User Computing (11:4), 3-13.

Guindon, R. (1990a) 'Designing the design process: Exploiting opportunistic thoughts', Human-Computer Interaction, No. 5

Guindon, R. (1990b) ‘Knowledge Exploited By Experts During Software System Design’, International Journal of Man-Machine Studies, Vol. 33, October 1990, pp 279-304

Heidegger, M. (1962) Being and Time, Harper & Row, New York

Highsmith, J. (2000) Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, New York, NY: Dorset House Publishing.

Hoc, J.M. , Green, T.R.G., Samurçay, R and Gilmore, D.J. (1980) [Eds.] Psychology of Programming, Academic Press, London, UK

Howcroft, D. and Wilson, M. (2003) "Participation: 'Bounded freedom' or hidden constraints on user involvement," New Technology, Work, and Employment. (18:1), 2-19.

Hutchins, E. (1991), ‘The Social Organization of Distributed Cognition’, in Lauren B. Resnick, John M. Levine, and Stephanie D. Teasley (eds.) Perspectives on Socially Shared Cognition, Washington, DC: American Psychological Association. pp. 283-307.

Jeffries, R., Turner, A.A., Polson, P.G. & Atwood, M.E. (1981) ‘The Processes Involved In Designing Software’, in J.R. Anderson (ed.) Cognitive Skills And Their Acquisition, Lawrence Erlbaum Associates, Hillsdale, New Jersey

Kant, E. and Newell, A. (1984) 'Problem solving techniques for the design of algorithms', Information Processing and Management, Vol. 28, pp 97-118

Kirsch, L.J. and Beath, C.M. (1996) "The enactments and consequences of token shared and compliant participation in information systems development," Accounting Management and Information Technology (6:4), 221-254.

Land, F. and Hirschheim, R. (1983) ‘Participative Systems Design: Rationale, Tools and Techniques’, Journal Of Applied Systems Analysis, Vol. 10.

Lanzara, G.F. (1983) ‘The Design Process: Frames, Metaphors And Games’, in U. Briefs, C. Ciborra, L. Schneider (eds.) Systems Design For, With and By The Users, North-Holland Publishing Company

Latour, B. (1987) Science in Action, Harvard University Press, Cambridge, MA

Latour, B (1991) ‘Technology is society made durable’, in J. Law (Ed.) A Sociology of Monsters. Essays on Power, Technology and Domination, Routledge, London, UK

Laukkanen, M. (1994) ‘Comparative Cause Mapping of Organizational Cognitions’, Organization Science, Vol. 5, No. 3, reprinted in J.R. Meindl, C. Stubbart, J.F. Porac (Eds.) Cognition Within and Between Organizations, Sage Publications, California, 1996

Lave, J. (1991) ‘Situating Learning In Communities of Practice’ in L.B. Resnick, J.M. Levine, S.D. Teasley (Eds.) Perspectives on Socially Shared Cognition, American Psychological Association, Washington DC, pp 63-82

Lave, J. & Wenger, E. (1991) Situated Learning: Legitimate Peripheral Participation, Cambridge University Press, Cambridge, UK

Law, J. & Callon, M. (1992) ‘The Life and Death of an Aircraft: A Network Analysis of Technical Change’ in W.E. Bijker and J. Law (Eds.) Shaping Technology/Building Society: Studies In Sociotechnical Change, MIT Press, Cambridge, MA

Lehaney, B., Clarke, S., Kimberlee, V., and Spencer-Matthews, S. (1999) "The Human Side of Information Development: A Case of an Intervention at a British Visitor Attraction.," Journal of End User Computing (11:4), 33-39.

Lewin, Kurt (1951). Field theory in social science: Selected theoretical papers (D. Cartwright, Ed.). Harper & Row, New York.

Mackenzie, D.A. & Wajcman, J. (1985) ‘Introduction’ to Mackenzie, D.A. & Wajcman, J. (Eds.) The Social Shaping Of Technology, Open University Press, Milton Keynes

Markus, M.L. and Bjorn-Andersen, N. (1987) "Power over users: its exercise by system professionals," Communications of the ACM June 1987 (30:6).

Mathiassen, L. & Stage, J. (1992) ‘The Principle of Limited Reductionism In Software Design’, Information Technology & People, Vol. 6, No. 2, pp 171-185

Mayer, R.E. (1989) ‘Human Nonadversary Problem-Solving’ in K.J. Gilhooley (Ed.) Human and Machine Problem-Solving, Plenum Press, New York

McCracken, D. D. & M. A. Jackson, (1982). Life Cycle Concept Considered Harmful. ACM SIGSOFT, Software Engineering Notes. 7(2):29-32

Mintzberg, H. & Waters, J.H. (1985) ‘Of Strategies, Deliberate and Emergent’, Strategic Management Journal, vol. 6, pp 257-72)

Mumford, E. (1983) Designing Participatively, Manchester Business School, UK

Mumford, E. & Sackman, H. (1975) Human Choice and Computers, North-Holland Publishing Company

Mumford, E. and Weir, M. (1979) Computer Systems in Work Design: the ETHICS Method, New York NY: John Wiley.

Necco, C.R. (1989) ‘Evaluating Methods of Systems Development: A Management Survey’, Journal of Information Systems Management, Vol. 6, Issue 1, pp 8-16

Nelson, D. (1993) "Aspects of Participatory Design - Commentary on Muller et al. 1993," Communications of the ACM (34:10), 17-18.

Nelson, E. (2002) "Extreme Programming vs. Interaction Design," FTP Online Magazine:January 15).

Preece, J., Rogers, Y., and Sharp, H. (2002) Interaction Design: Beyond Human-Computer Interaction, New York, NY: Wiley.

Rittel, H.W.J. (1972) ‘Second Generation Design Methods’ Reprinted in N. Cross (Ed.) Developments In Design Methodology (1984), J. Wiley & Sons, Chichester, pp 317-327

Rittel, H.W.J. & Webber, M.M. (1973) Dilemmas in a General Theory of Planning, Policy Sciences, Vol. 4, pp 155-169

Rosenbrock, H.H. (1988) ‘Engineering As An Art’, AI & Society, Vol. 2, No. 4, pp 315-320

Royce, W. W., (1970) 'Managing the Development of Large Software Systems: Concepts and Techniques', in Proceedings of WESCON August 1970, pp 1-9. Reprinted in: Proceedings ICSE 9, Computer Society Press, 1987, pp. 328-338.

Scarbrough, H. and Corbett, J.M. (1991) Technology and Organisation: Power, Meaning and Design, Routledge.

Schön, D.A. (1983) The Reflective Practitioner: How Professionals Think In Action, Basic Books, NY

Simon, H. A. (1957) Models of Man: Social and Rational, New York: John Wiley.

Simon, H.A. (1960) The New Science of Management Decision, Harper & Row, New York

Simon, H.A. (1973) ‘The Structure of Ill-Structured Problems’, Artificial Intelligence, No. 4, pp 145-180

Simon, H.A. (1981) Sciences of The Artificial, Second Edition, MIT Press, Cambridge, MA

Smircich, L. & Morgan, G. (1982) ‘Leadership: The management of meaning’, Journal of Applied Behavioural Science, Vol. 18, No. 3, pp 257-273

Star, S. L. (1989), ‘The Structure of Ill-Structured Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving’, in L. Gasser and M. N. Huhns (eds.) Distributed Artificial Intelligence, Vol. II. San Mateo, CA: Morgan Kaufmann Publishers Inc., pp. 37-54.

Suchman, L. (1987) Plans And Situated Action, Cambridge University Press, MA, USA

Sumner, M. & Sitek, J. (1986) ‘Are Structured Methods for Systems Analysis and Design Being Used?’, Journal of Systems Management, June 1986, Vol. 37, Issue 6, pp 18-23

Taylor, F.W. (1911) Principles Of Scientific Management, Harper, New York

Visser, W. & Hoc J-M. (1990) ‘Expert Software Design Strategies’ in J.M. Hoc, T.R.G. Green, R. Samurçay, D.J. Gilmore (Eds.) Psychology of Programming, Academic Press, London, UK

Weber, M. (1922), The Theory of Social and Economic Organization, translated by A.M. Henderson and T. Parsons (Eds.), Oxford University Press, New York NY, [This English translation published 1947.]

Winograd, T. & Flores, F. (1986) Understanding Computers And Cognition, Ablex Corporation, Norwood, New Jersey

Winograd, T.A. (1994) "Designing a language for interactions," interactions (1:2), 7-9.

Winograd, T. (1996) "Introduction," In: T. Winograd (ed.) Bringing Design To Software, New York NY: Addison-Wesley Publishing.

Winter, M.C., Brown, D.H. & Checkland, P.B. (1995) ‘A Role For Soft Systems in Information Systems