JITTA
JOURNAL OF INFORMATION TECHNOLOGY THEORY AND APPLICATION

HUMAN-CENTERED VS. USER-CENTERED APPROACHES TO INFORMATION SYSTEM DESIGN

SUSAN GASSON, **Drexel University**
College of Information Science and Technology, Philadelphia, PA, USA, Email: sgasson@cis.Drexel.edu

ABSTRACT

Despite continuing debates about the "user" emphasis in HCI, new design approaches, such as interaction design, continue to focus on humans as technology users, constraining the human-centeredness of design outcomes. This paper argues that the difference between "user" focus and a human-centered focus lies in the way in which technology is designed. The emphasis on problem closure that is embedded in current approaches to designing information systems (IS) precludes an examination of those issues central to human-centered design.

The paper reviews recent approaches to user-centered IS design and concludes that these methods are targeted at the closure of technology-centered problems, rather than the investigation of suitable changes to a system of human-activity supported by technology. A dual-cycle model of human-centered design is presented, that balances systemic inquiry methods with human-centered implementation methods. The paper concludes with a suggestion that IS design should be viewed as a dialectic between organizational problem inquiry and the implementation of business process change and technical solutions.

INTRODUCTION

By focusing on usability, the IS literature too often overlooks the social context of use. Bjorn-Andersen (1988) criticized the narrow definition of human-computer interaction (HCI) in the literature, with the words: "it is essential that we see our field of investigation in a broader context. A 'human' is much more than eye and finger movements". So how do we design for human-centeredness? 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." The human-centered approach to the design of technology arose as

Ken Peffers acted as senior editor for this paper.

Gasson, S., "Human-Centered Vs. User-Centered Approaches to Information System Design", *The Journal of Information Technology Theory and Application (JITTA)*, 5:2, 2003, 29-46.

a reaction to perceptions that traditional approaches to technology design deskill technology users and impoverish the quality of working life (Gill, 1991; Scarbrough and Corbett, 1991). While many of the issues of human-centeredness have been adopted by the IS and HCI literature, many have been considered to lie outside the boundaries of "user" interactions with computers. This is because of a focus on technology and how humans interact with technology, rather than questioning how and why technology may be of service in supporting human work. Despite continuing debates about a focus on human actors as "users" of technology, this issue has not gone away and continues to constrain new, "user-centered" approaches to IS design, such as agile software development (Beck, 1999; Fowler and Highsmith, 2001; Highsmith, 2000) and interaction design (Cooper, 1999; Preece, Rogers and Sharp, 2002; Winograd, 1994). These constraints sit poorly with the need to design systems that support emerging knowledge processes (Markus, Majchrzak and Gasser, 2002) and result in systems that do not support the processes required to support organizational work (Butler and Fitzgerald, 2001; Lehaney, Clarke, Kimberlee and Spencer-Matthews, 1999).

This paper is structured as follows. The next section provides a discussion of the tenets of human-centered design and why this is not catered for in the mutual adaptation that is theorized to take place between organization and technology. Then we examine what we know about the nature of IS design processes, that makes human-centeredness problematic. Following this, the paper critiques some recent developments in IS design, from the perspective of human-centeredness:

- Participatory design is discussed as an alternative to the traditional, technology-centered system development life-cycle that resulted from an emphasis on human-computer interaction (HCI).
- Interaction design, a development of HCI that considers work processes is examined.
- Use-cases as part of a Unified Modeling Language (UML) approach are discussed, as a recent advancement for modeling business processes and user-interactions with the intended information system.
- Agile Software Development is presented, as uniquely a practitioner-initiated approach to human-centeredness in IS design.

The paper argues that each of these approaches focuses on user-centeredness at the expense of human-centeredness, because of an implicit IS focus on technical problem-closure, rather than inquiry. An alternative, "dual cycle" model of IS design is presented, that focuses on problem definition jointly with problem closure, based on a longitudinal study of stakeholder design.

HUMAN-CENTERED INFORMATION SYSTEM DESIGN

Recent theories that explain the relationship between technology and organization have argued that the two are mutually interdependent: each shapes the other

CONTRIBUTION

The main contribution of this paper is to argue that "user-centered" system development methods fail to promote human interests because of a goal-directed focus on the closure of predetermined, technical problems.

The paper is unusual, in that it questions the traditional interpretation of human-centeredness found in the HCI and IS literatures, as the production of a usable system design. The author critiques a number of recent developments in human-centered design methods, to examine the extent to which their focus on stakeholders as simply users of technology limits the extent to which they can support organizational work.

Finally, the paper presents a "dual-cycle" design model, that balances technical problem closure with organizational problem inquiry. The need for a dialectic process, to achieve a balance between human-centered system outcomes and the design of an effective, formal technical IS solution is emphasized.

through self-reinforcing cycles of sensemaking and giving form to the organizational meanings that ensue (Majchrzak, Rice, Malhotra, King and Ba, 2000; Orlikowski, 1992; e.g. Orlikowski, 2000; Scarbrough and Corbett, 1991). But the process by which meanings are explored and then translated into organizational procedures, with their supporting technical artifacts – the process of design – has received relatively little attention. Information technology (IT) is most often viewed as a "black box", the form of which is predetermined by decisions as to its role and purpose (Orlikowski and Iacono, 2001). But the physical ways in which users may interact with an IT system, the work-processes that are supported or not supported, and the extent to which users are permitted to control IT system processes fundamentally affect how work is performed, regardless of the adaptation processes that follow. For example, in a study of computer-supported factory automation, Wilkinson (1983) reports that a company which wanted to purchase a system that permitted their shop-floor workers to control the manufacturing process found that there were none available on the market. The designers of such systems assumed a managerial intention to remove autonomy from manufacturing workers and so designed systems to prevent workers from "tampering" with production control parameters. Similarly, Button et al. (Button, Mason and Sharrock, 2003), writing twenty years later, discuss how a workflow and information management system prevented workers from managing their work in the most effective way, because of assumptions built into the system about the flow of work. The need to understand a "web" of computer-supported activity, when designing an information system (Kling and Scacchi, 1982) and to understand how organizational purposes are transformed through the IS design and implementation process (Markus and Bjorn-Andersen, 1987; Markus and Robey, 1988) appear to be well established principles in the IS literature. Yet these principles appear to have had relatively little impact on IS research or practice (Orlikowski and Iacono, 2001). These issues have largely disappeared from the IS literature. As a result, there are few papers that do not uncritically adopt the HCI perspective that human-centeredness = usability. Much of the work that deals with human-centeredness is dated, or is located in the organizational management literature; this is reflected in the discussion here.

Human-centered design takes a socio-technical view (Emery and Trist, 1960), balancing the requirements of two, competing "systems" (Hedberg and Mumford, 1975; Heller, 1989):

- The social system of interacting human activities, multiple, implicit (and often conflicting) goals, human understanding and knowledge, business context and application-specific cultures and practice.
- The technical system of formal, rule-based procedures and technology, managed by performance indicators and exception-handling.

The difficulties inherent in achieving this balance have been recognized in organizational literature on the impact of technological change at work. A human-centered approach takes the design 'problem' from work-participants – this is often embedded in local, organizational practice, rather than seeking a technical solution to a context-free, information-processing problem (Lehaney, Clarke, Kimberlee and Spencer-Matthews, 1999). The main tenets of this, "human-centered design" perspective are:

1. Human-centered design advocates the design of flexible systems that permit the people who work with them to shape and manage their work (Gill, 1991; Kapor, 1996; Lehaney, Clarke, Kimberlee and Spencer-Matthews, 1999).
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, 1999). Human-centered design advocates the design of systems that question normative expectations of technology (Kuhn, 1996).
3. The human-centered approach is opposed to the traditional, technology-oriented approach, which prioritizes computer-based information processing and

technology-mediated communications over humans and their communicative collaboration (Barthélemy, Bisdorff and Coppin, 2002; Gill, 1991).

4. Human-centered systems 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 (Kuhn, 1996; Lehaney, Clarke, Kimberlee and Spencer-Matthews, 1999).
5. The explicit, rule-based knowledge needed for computer-based systems is useless without the tacit and skill-based knowledge through which explicit knowledge is filtered (Cooley, 1987; Rosenbrock, 1988). Human-centered design acknowledges the need for *informal* information systems that enable the use and communication of implicit knowledge (Land, 1992).
6. We should avoid the prevailing tendency to separate "planning" tasks from "doing" tasks, as this separation results in deskilled technology users who are ill-equipped for exception-handling or meaningful decision-making (Cooley, 1987). Human-centered design strives for socio-technical systems that support meaningful, enriched work (Gill, 1991; Lehaney, Clarke, Kimberlee and Spencer-Matthews, 1999).

If nothing else, human-centered design is predicated on enlightened self-interest. Technologies are designed around a set of assumptions concerning what work processes are required and how they will take place that are often simply wrong (Button, Mason and Sharrock, 2003; Dourish and Button, 1998). A technology focus fails to take into account the distributed and informal nature of expertise and decision criteria (Barthélemy, Bisdorff and Coppin, 2002; Land, 1992).

Stakeholder interpretations of organizational processes, goals and needs may differ considerably, depending on the work or interest-group to which they belong (Gasson, 1999b; Lave, 1991; Weick, 1979). The requirements for an information system are located in multiple "communities of practice": groups of people who work together to achieve specific ends, in locally-defined ways (Boland and Tenkasi, 1995; Wenger, 1998). Knowledge about how to perform work processes and the role that an information system might play in the organization is often implicit and difficult to communicate (Brown and Duguid, 1992).

Individuals inhabit a socially constructed world and through their actions, reproduce and give meaning to that world (Berger and Luckman, 1966; Kelly, 1955; Weick, 1979, 2001). People create a personal system of psychological constructs, which varies as they successively construe replications of events (Kelly, 1955). Through the use of specific social genres and forms of communication, individuals not only pursue their goals, but they define a situation and a problem at hand, they present themselves to the external world and they recreate personal and group identities (Habermas, 1987; Strauss, 1983; Yates and Orlikowski, 2002). People shape and are shaped by this experienced "lifeworld" (Habermas, 1987) and that in turn shapes how they conceptualize an organizational information system. Thus, an organizationally-situated design is the result of negotiation between multiple, social “worlds”, that represent reality in different ways (Strauss, 1983). The resulting IS reflects intersections between an overlapping set of individual and group perspectives, that shift and evolve as the design proceeds.

The notion that design is driven by a consensual set of goals, determined at the start of the analysis, is a vast over-simplification. Goal-directed methods, that do not revisit the initial goals for a problem solution, but take these as given throughout the design, lose the opportunity to benefit from the learning that accrues through the process of design and may be subject to implicit goal-redefinition. For example, in a study by Gasson (1999a), a user-centered design project failed because of the different ways in which non-technical and technical design participants communicated and evaluated the knowledge about the design. The legitimacy of certain design goals was judged differently by the two subgroups participating in the project and this affected

which goals were acted upon by different subgroups. Through their ability to control the technical implementation of the design, the technical developers subverted the original intentions of the project to a considerable extent. But the formal goals for the project remained unchanged.

An analysis of technology as the malleable product of improvisational adaptation (Lau, Doze, Vincent, Wilson, Noseworthy, Hayward and Penn, 1999; Orlikowski and Hofman, 1997) ignores the technological decision-making process that gives the artifact a specific form. During this process, new purposes and roles for the technology emerge, are debated, and may replace the original purposes, as more technically "appropriate" (Gasson, 1999a). Many assumptions and interests are unreflectively embedded into the technical artifact during system design, that constrain its potential role and use in organizational work (Akrich, 1992; Mackenzie and Wajcman, 1999; Xu, Lehaney, Clarke and Yanqing, 2003). Refrigerators hum because everyone knows that refrigerators hum … no-one questions the use of the specific coolant circulation technology that makes them hum. Similarly, computer-based information systems are configured in a certain way because it makes technical sense to locate functions in a certain way. No-one questions the impact that this will have upon how people can use the system, until these decisions have been made and the designers turn their attention to making the system "usable". But, by then, these decisions may fundamentally constrain the ways in which people can perform their work processes (Button, Mason and Sharrock, 2003; Gasson, 1999a). Given that many of these ideas have been debated and explored for several years, the question arises as to why these ideas have not been incorporated into the design of IT systems. The next section discusses the nature of the design process, to explore this question.

THE NATURE OF THE IS DESIGN PROCESS

Traditionally, IS design is viewed as a single stage in the systems development life-cycle (SDLC), defining a detailed physical form for the technical component of an information system. The traditional SDLC (the waterfall model) has three main limitations as a guide to the design of organizational information systems. Firstly, it relates to the development of systems to support relatively well-defined, technical goals and tells us little about how ill-defined and unbounded problems should be defined and resolved (Lanzara, 1983; Mathiassen and Stage, 1992; Rittel, 1972). Secondly, it is based on a model of design as individual, rational problem-solving, whereas organizational IS design tends to involve collaborative action, situated in a social and political context that is far from rational (Boland and Tenkasi, 1995; Preston, 1991). Thirdly, it assumes that objective goals and solution requirements may be defined early in the design process whereas empirical research tells us that IS goals emerge through the processes of design and that these goals are political, subjective and negotiated (Boland and Day, 1989; Gasson, 1998; Guindon, 1990). However, the assumptions of the waterfall model appear to underlie many current approaches to IS design. For example, the recent revival of interest in "pattern languages" is based on the concept that inherent patterns exist in organizations and so computer-supported activity may be "ordered" by designing them according to "some of the physical structures that make an environment nurturing for human beings" (Alexander, 1999, page 73). Much of the appeal of the Unified Modeling Language (UML) approach (Booch, Rumbaugh and Jacobson, 1996), discussed below, appears to rest upon its goal-directed (and therefore decompositional) nature, making the production of systems and software easier to manage and control.

Simon's (1960; 1973) assumption of goal-directed (and thus objectively justifiable) behavior in design have been adopted widely: this assumption has received remarkably little attention in the IS literature (Checkland and Holwell, 1998). This is perhaps because the result of challenging this perspective is to conclude that design is not amenable to planning, in the manner previously thought. Simon (1973) argues that ill-structured problems, such as the design of organizational information systems, are associated with a consensual set of goals for the solution of an

objectively-defined problem. Thus, a solution can be derived rationally (or at least, in ways that may be justified on the basis of rationality), from these goals. But empirical research into "expert" software and IS design has demonstrated that design strategies are "improvisational" (Lau, Doze, Vincent, Wilson, Noseworthy, Hayward and Penn, 1999; Orlikowski, 1996; Orlikowski and Hofman, 1997; Weick, 2001) or "opportunistic" (Ball and Ormerod, 1995; Guindon, 1990; Khushalani, Smith and Howard, 1994), in practice. Individuals appear to be guided by locally-contingent and partial plans (Majchrzak, Rice, Malhotra, King and Ba, 2000; Malhotra, Thomas, Carroll and Miller, 1980; Suchman, 1987; Turner, 1987), to resolve problems that are subjectively-defined, are interrelated with other problems and are amenable to many, often incompatible solutions (Ackoff, 1974; Rittel, 1972). Problem definition is guided by the designer's experience of, or exposure to, suitable complete or partial solutions (Majchrzak, Rice, Malhotra, King and Ba, 2000; Malhotra, Thomas, Carroll and Miller, 1980; Turner, 1987). Problem and solution are conceived together and are inextricably intertwined (Bansler and Bødker, 1993).

These insights demonstrate that we need to view IS design as involving problem-exploration jointly with problem closure. Taking this approach would allow us to continually examine decisions concerning the role of IT within an organizational “system” of work (Checkland and Holwell, 1998). The remaining sections of this paper examine “state of the art” approaches to IS design that purport to focus explicitly on human-centeredness, to examine the extent to which these approaches support a problem-centered focus.

DEVELOPMENTS IN HUMAN-CENTERED DESIGN APPROACHES

Participatory Design

The 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" problem discussed above. 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 and Weir, 1979; Mumford, 1983) 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, Clarke, Kimberlee and Spencer-Matthews, 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, and 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 exercise *conceptual* power, in managing 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, 1999a). 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, 1999a; 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, Rogers and Sharp, 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, pages 8-9).

So interaction design has the potential to consider a "space of possibilities", that encompasses many different and subjective definitions of the organizational problem. A human-centered solution would be one that negotiates the needs of the multiple stakeholder "worlds". This would lead to many alternative technical solutions being evaluated by stakeholders. But in practice, interaction design appears to be limited by the tradition of HCI discourse. It examines how a single user might use a predefined technical artifact, to determine how to design the artifact to be usable. As Cooper (1999) argues, analyzing how people might want to use an artifact is a significant advance over current methods of design. Cooper (1999), who claims to have invented the approach, defines interaction design as "goal-directed design" that 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 and to cope with this, a

human-centered design approach needs mechanisms for eliciting and capturing goals that emerge as the design proceeds.

A similar, goal-driven approach is taken by Preece et al. (2002), who emphasize "the interactive aspects of a product" (*page 11*). Although they emphasize the evolutionary nature of design and extend the goal-driven concept with rich discussions of use, their perspective is also essentially driven by the notion that design is centered around the 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.

The discourse of Interaction Design *starts* with a concept of "the computer" (or computer-based technology) and only then considers the context of the human-computer interaction. This has the effect of moving the design model back to the historically *unitary* focus of HCI: a single technology user, moving towards closure of a single, task-related problem, in isolation from the social world of work that surrounds them. Interaction is thus reduced to interface.

HCI research into user-centered design has, however, had a significant impact on software development *practice*, as evidenced by the emergence of two schools of design methodology: the production of Use-Cases as part of Uniform Modeling Language (UML) approaches to system design and "agile" software development.

Use-Cases in UML

The production of use-cases (Jacobson, 1991) has now been absorbed into the Unified Modeling Language (Booch, Rumbaugh and Jacobson, 1996) approach to formal system representation and modeling. The primary concerns here are the correctness and completeness of a technical system model. Use-cases constitute a representation of *interactions* between different classes of user and a computer system. From these use-cases, formal object-oriented models and specifications may be defined, that enable the production of a technical system. A example of a use-case diagram is shown in Figure 1. Special cases (extensions) of associations between objects or business processes are shown with a dotted line, while normal associations are shown with a solid line. So in the example given here, the credit limit would only be checked in some circumstances (e.g. if the account balance is insufficient for the withdrawal).

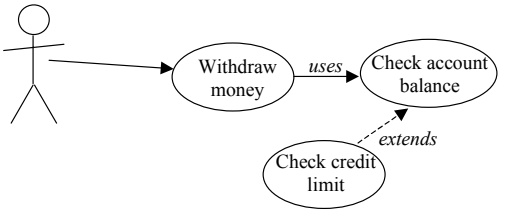


Figure 1: Example of a Use-Case Model

A benefit of the method is that designers are encouraged to base use-case models on the viewpoints of, and interactions between, multiple stakeholders. A high-level model is thus constructed that, ostensibly, starts with the user requirements of the proposed system and develops a set of "business" processes that the system will (partly or wholly) automate.

In HCI terms, this method represents a major victory for user-centeredness in technical practice. System conceptualization starts with an understanding and definitions of user-interactions. But is this really true? The use-case model focuses on the articulated requirements of a single user. It has no way of surfacing (or even recognizing the existence of) implicit requirements. Where multiple viewpoints are sought, the task is to reconcile these, not to *represent* (often conflicting) user requirements of the system. Additionally, most use-cases appear to be produced by the designer imagining how users would interact with their target system, to derive a set of algorithmic business-rules. Use-cases are largely based on short interviews and there is little opportunity for validation. So we revert to the problem with traditional approaches, identified by Norman (1990):

> "The designer expects the user's model to be identical to the design model. But the designer does not talk directly with the user - all communication takes place through the system image." (Norman,

1990, page 16).

The most serious problem with the UML approach (from a human-centered perspective) is that the design of user-interaction starts with a concept of the computer system, that is not challenged by the process of design inquiry. Designers gradually build up a set of interactions with a technical system whose form is preconceived (as demonstrated by their ability to imagine and represent interactions with this system), rather than with a conceptualization of users, their work, the problems that they face, and their lifeworlds. An understanding and definition of system interactions are therefore formed by the designer constructing a model of the system as a set of functions, rather than by an understanding of the needs of potential users and other system stakeholders. So, while system design based on use-cases may be considered user-centered, it does not fulfill the requirements for human-centeredness. This approach focuses on interactions with a technical system, ignoring the wider social context and the emergent, negotiated purposes of the system.

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, Gray and Seewalt, 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 (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.

The Need To View Human-Centered Design As Mutually-Interacting Inquiry and Implementation

I have argued here that the IS and HCI literatures have largely ignored the effect that the *forms* of available technology have, upon the range of social choices available and the role that IT systems play in work design. I examined a number of developments in "human-centered" IS design, to determine the extent to which they could achieve those elements of a human-centered outcome that were defined above. The findings are summarized in Table 1.

Table 1. Summary of Human-Centered IS Design Approaches

Approach	Intended Focus	Actual Focus
Traditional IS design approaches	The structuring of ill-structured problems: goal-driven decomposition.	Explicit (management) focus is goal-driven and decompositional. Implicit strategies are opportunistic, to deal with goal-emergence.
Prototyping and participatory design	Negotiation and exploration of ill-structured problems.	Iterative and cyclical process of stakeholder involvement, limited by political selection of user-representatives and technology-centered requirements focus.
Interaction design	Exploration of IT-supported user work-processes.	Technology-centered, individual user focus. Assumes consensus among system users, with well-understood IS goals.
UML and Use-Cases	Modeling of business processes and user-interactions with intended IT system.	Models formal information-processing (business processing rules). Technology-centered and decompositional (so no opportunity to redefine goals as these emerge through design process).
Agile Software Development	Adaptation of an evolving system design, based on user interaction and scenario generation.	Technology centered prototyping, accomplished by the development of individual user-scenarios.

Most of the more recent approaches are iterative and so avoid the problems of the traditional, decompositional focus. But I would argue each of these approaches focuses on user-centeredness at the expense of human-centeredness, because of their technology-centered focus. To embrace the tenets of human-centeredness discussed earlier in this paper, system stakeholders -- the intended "victims and beneficiaries" of the proposed information system (Checkland, 1981) -- should be enabled to negotiate the role and purposes of the system with other stakeholders, non-technical as well as technical. This is not a new idea - it was proposed by Mumford (1983), early in the participative design movement. But its implementation has been problematic, because of the persistence of the goal-driven, technology focus in IS design. It can be seen from the discussion here that most user-centered approaches are concerned with *closing down* a technology-centered and goal-directed IS problem-definition, not about *exposing* (or opening up) the social and organizational context (the design "problem") to examination and debate. Even in agile software development, the design problem is determined very early in the process and remains unexamined after that point. The user-interaction and use of scenarios may generate new IT system requirements-goals, but it does not question the essential form and social role of the technical system, as even this approach focuses on an individual "user" of technology. The most human-centered of the methods discussed, participatory design approaches, do not change the fundamental nature of the "circular" system development life-cycle. They merely "rotate" the life-cycle through 90°, so that the cycle is driven by user-evaluation of system design requirements, rather than by technical evaluation of system design requirements. This rotation does not question many of the essential contradictions of the traditional perspective, because it inherits the "problem closure" life-cycle emphasis.

To resolve these problems, design may be managed as shown in Figure 2. This model is based on findings from a longitudinal study of a stakeholder-driven IS design in a midsize engineering company (Gasson, 1998). A group of stakeholders was suggested by the various organizational groups who would be affected by the proposed system. These stakeholders met regularly, to discuss how work processes should change and how the new IT system should be defined, to support these processes. derived from this study and from other case studies of stakeholder-driven design. The model has been refined according to the findings of other studies (Gasson and Holland, 1996; Gasson, 1999a) and ongoing case investigations. It therefore represents an optimal way of managing the dialectic between subjective, organizational problem inquiry and goal-directed, process and technical solution design.

The model in Figure 2 represents two "cycles" of the design process, to deal separately but interactively with system inquiry and implementation (opening-up of organizational problems and closing-down of business process/technical solutions).

The first iteration is the cycle of inquiry, in which organizational "problems" are debated, negotiated and defined:

1. Stakeholders agree a set of explicit goals for the organizational and information system changes.
2. Based on these goals, a single stakeholder, who is familiar with a particular area of work, produces a "paper prototype" process design to achieve the goals for change. As part of this prototype, they conceptualize the role that a computer-based IS should play in this process. Other group members critique the suggested design: both the process changes and the IT system concept are refined, as part of this process.
3. This is followed by a definition and validation stage, where the group defines and agrees deliverables, in the form of organizational process changes and IT system goals. This is also the transition-point to the implementation cycle (stage 5), if the group feels that the goals and requirements for change are clear and consensual enough for them to proceed.

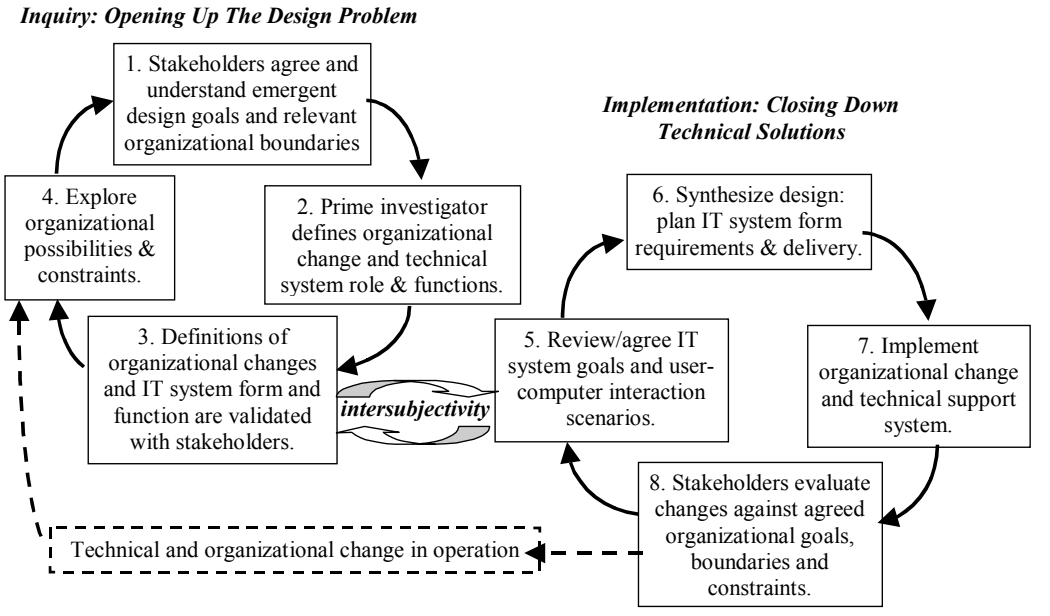


Figure 2. A Dual-Cycle Model Of Human-Centered Design (Adapted from Gasson, 1998)

4. If the group feels that the changes are still not well-understood (or not understood by group members in common), they explore organizational possibilities and constraints that operate upon the process, from the perspective of their domain-worldview. Revised goals for change are suggested through this process of "argumentation", that leads back to the goal-definition activity (1).

As new information emerges and individual stakeholders understand interactions between their own work domains and those of others, goals are redefined, often in small but significant ways that redefine the role of the computer-based IS. For example, in the study from which this model was initially derived, the definition of the computer-based IS as a way to track and chase individuals to complete sections of a document was quietly dropped, as stakeholders agreed that they did not want "this big snake that runs through the whole company". In the first few iterations, stakeholders understood the goals and suggested solutions in different ways -- a reflection of the different "worlds" discussed above. But as they iterated around this cycle, they developed a degree of a shared understanding, based on the negotiation and development of a common worldview, that enabled them to debate design issues more meaningfully than before. At the validation stage, they reached a clear agreement that this design would satisfy their needs for change. This intersubjectivity permitted them to move to the implementation cycle.

IS solutions are defined through the implementation cycle, which is driven by the goals and problem-definitions defined in the inquiry cycle:

5. The implementation cycle starts with a review of goals and scenarios, based on the previous cycle's definition of the organizational problem. This activity ensures that goals and requirements for the IS-related change are clearly defined and their implications for the solution are well-understood.
6. Stage 5 produces an explicit and agreed set of changes that are synthesized into a set of requirements and deliverables by what is now a fairly knowledgeable set of stakeholders -- at least in terms of the part of the business process for which a solution has been agreed so far. It is at this point that the physical form of the IT system is agreed (e.g. "we need a data portal to

deliver these documents").

7. As the group now shares a common understanding of the goals and requirements for change, the implementation of organizational changes and the computer-based IS may be delegated to the oversight of individual stakeholders. For example, in the longitudinal study, other stakeholders cheerfully delegated the development of the technical system to the IT manager and delegated the implementation of the initial process changes to the manager of the existing process.
8. Once the changes have been made, stakeholders can evaluate them against their agreed goals for change, with a common understanding of the organizational goals and constraints within which they operate. This understanding is communicated to stakeholders' local workgroups, which also achieves a higher sense of shared ownership than is normally the case. The evaluation results either in the process and IS changes becoming operational, or the group may feel that a formal review of the evaluation is needed, to debate problems that need to be dealt with in another cycle of the design, or changes to this or another process. In this case, the group moves back to the cycle of inquiry (stage 3), defining what elements need to be dealt with next. The process ends when the group feel that all significant areas of change have been dealt with and the new system of business processes and IS is operating adequately.

The dual-cycle model emphasizes the importance of systemic inquiry as part of the design process. The problem closure models within which we normally work delegitimize user participation and prevent users from revisiting problem definitions (Gasson, 1998, 1999a). This model acknowledges the process of involving stakeholders in defining actions for organizational and technical change, as part of a procedural design approach. Although the implementation stage is treated almost superficially by this model, this is not to claim that the implementation of technical or organizational change is unproblematic. We already have good methods to manage this stage, resulting from HCI research and practice, many of which are discussed above. However, problem inquiry and definition are insufficiently researched and insufficiently interrelated, in the way in which we approach system design. This model presents an alternative to the limited models of technical problem-closure that are implicit in each of the approaches discussed above.

CONCLUSIONS AND IMPLICATIONS FOR PRACTICE

In fields such as architecture, design is viewed holistically, as the synthesis of problem exploration and solution definition (Lawson, 1990; Winograd, 1996). But the IS perspective takes a view of human agency that reduces human-centeredness to those considerations required to model individual interactions with a computer-system. It thus avoids considerations relating to emancipation, autonomy, and the role of IT configuration in enabling or constraining organizational work. The difference between a "user" focus and a human-centered focus lies in the way in which technology is designed. This paper argued that "user-centered" system development methods fail to promote human interests because of a goal-directed focus on the closure of predetermined, technical problems. The traditional interpretation of human-centeredness as the production of a usable system design, found in the HCI and IS literatures, was found wanting. A number of recent developments in human-centered design methods were examined. It was argued that their focus on stakeholders as simply users of technology limits the extent to which these methods can support effective, human-centered organizational work. Finally, the paper presented a "dual-cycle" design model, that balances technical problem closure with organizational problem inquiry. The need for a dialectical process, to achieve a balance between human-centered system outcomes and the design of an effective, formal technical IS solution was emphasized.

I have argued that new, "user" centered IS design approaches are as limited as traditional approaches to IS design, because of their emphasis on problem closure. But sometimes, a goal-oriented approach *is*

appropriate. If stakeholders can agree a well-defined problem, which can be solved with known technology, goals are relatively easy to set. The decision about which design method(s) to use depends on the familiarity of stakeholders with the proposed type of information system technology, and on the ease with which the IS "problem" can be negotiated and agreed among stakeholders. If stakeholders feel that the organizational "problem" is well-defined and agreed, then "user-centered" methods are appropriate, for the design of that part of the solution which is computer-based. If stakeholders cannot agree a suitable problem definition, then complex inquiry methods are required to provide such a definition, which needs to be revisited periodically, as suggested by the dual-cycle model of Figure 2.

In the spirit of inquiry, we conclude with a problem-statement and a proposed solution.

It is proposed that the two "worlds" of socio-cultural work and technology-interaction are incommensurable. We cannot analyze them using common methods, nor can we derive procedures and methods for producing software that satisfies the needs of both. HCI methods are targeted at closing-down technology-*centered* problems, rather than opening up a technology-*supported* system of human-activity for examination and change. In "user-centered" system design approaches, the boundary of inquiry is too limited for designers to consider those aspects of context and socio-cultural significance that would make the system "human-centered".

Recent developments in "user-centered" system design do not engage with the core problems of traditional systems development: its focus on technical problem-closure and its view of system stakeholders as either "managers" (and therefore definers) or "users" (and therefore consumers) of IT. This means that there is no mechanism by which emergent goals and requirements for IS-related change can be made explicit, so they can be debated among affected stakeholders. What is required is a combination of systemic inquiry methods with user-centered interaction methods for the design of supporting technology, with a dialectic between the two. We should focus on separating problem investigation from solution design, using approaches and methods that permit us to operate in different modes of inquiry in each world. We can then use that unique, human quality that we all possess - the ability to synthesize across incompatible domains of knowledge - to produce appropriate and human-centered solutions for the real world.

REFERENCES

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

Akrich, M. "The De-Scription Of Technical Objects," in: Bijker, W.E. and Law, J. (eds.), Shaping Technology/Building Society: Studies In Sociotechnical Change, MIT Press, Cambridge, MA, 1992, pp. 205-224.

Alexander, C. "The origins of pattern theory: The future of the theory and the generation of a living world," IEEE Software, 1999, 16:5, pp. 71-82.

Ball, L.J., and Ormerod, T.C. "Structured and opportunistic processing in design: A critical discussion," Int. Journal of Human-Computer Interaction, 1995, 43:1, pp. 131-151.

Bansler, J.P., and Bødker, K. "A Reappraisal of structured analysis: design in an organizational context," ACM Transactions on Information Systems, 1993, 11:2, pp. 165-193.

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

Barthélemy, J.P., Bisdorff, R., and Coppin, G. "Human centered processes and decision support systems," European Journal of Operational Research, 2002, 136:2, pp. 233-252.

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

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

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

Bjorn-Andersen, N. "Are Human Factors Human?," The Computer Journal, 1988, 31:5, pp. 386-390.

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

Boland, R., and Day, W.F. "The experience of systems design: a hermeneutic of organisational action," Scandinavian Journal of Management, 1989, 52:2, pp. 87-104.

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

Booch, G., Rumbaugh, J., and Jacobson, I. Unified Modeling Language User Guide, Addison-Wesley, Reading, MA, 1996.

Brown, J.S., and Duguid, P. "Enacting Design for the Workplace," in: Adler, P.S. and Winograd, T.A. (eds.), Usability: Turning Technologies Into Tools, ACM Press, New York, NY, 1992, pp. 164-197.

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

Button, G., Mason, D., and Sharrock, W. "Disempowerment and Resistance In The Print Industry? Reactions To Surveillance-Capable Technology," New Technology, Work, and Employment., 2003, 18:1, pp. 50-61.

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

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

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

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

Cooley, M. Architect Or Bee?: The Human Price Of Technology, Hogarth Press London, 1987.

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

Dourish, P., and Button, G. "On "Technomethodology": Foundational Relationships between Ethnomethodology and System Design," Human-Computer Interaction, 1998, 13:4, pp. 395-432.

Emery, F.E., and Trist, E.L. "Socio-Technical Systems," in: Churchman, C.W. and Verhulst, M. (eds.), Management Science Models and Techniques, Pergamon Press, Oxford UK, 1960.

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

Fowler, M., and Highsmith, J. "The Agile Manifesto," Software Development, 2001, August, pp. 28-32, Available online at: http://www.sdmagazine.com.

Gasson, S. "Framing Design: A Social Process View of Information System Development," Proceedings of The Nineteenth International Conference on Information Systems (ICIS 98), Helsinki Finland, 1998.

Gasson, S. "A Social Action Model of Information Systems Development," The Data Base For Advances In Information Systems, 1999a, 30:2, pp. 82-97.

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

Gasson, S., and Holland, N. "The Nature And Processes Of IT-Related Change," in: Orlikowski, W.J., Walsham, G., Jones, M.R. and deGross, J.I. (eds.), Information Technology and Changes in Organizational Work, Chapman & Hall, London, 1996.

Gill, K.S. "Summary Of Human-Centred Systems Research In Europe, Part 1," Systemist, the journal of the UK Systems Society, 1991, 13:1, pp. 7-27.

Guindon, R. "Designing the design process: Exploiting opportunistic thoughts," Human-Computer Interaction, 1990, 5, pp. 305-344.

Habermas, J. The Theory of Communicative Action - Critique of Functionalist Reason, Beacon Press, Boston, MA, 1987.

Hedberg, B., and Mumford, E. "The Design Of Computer Systems: Mans Vision Of Man As An Integral Part Of The Systems Design Process," in: Mumford, E. and Sackman, H. (eds.), Human Choice And Computers, North-Holland, Amsterdam., 1975, pp. 31-59.

Heller, F. "Human Resource Management And The Socio-Technical Approach in New Technology," in: Bamber, G. and Lansbury, R.D. (eds.), New Technology: International Perspectives On Human Resources And Industrial Relations, Unwin Hyman Ltd., London, 1989.

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

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

Jacobson, I. Object-Oriented Software Engineering, ACM Press, 1991.

Kapor, M. "A Software Design Manifesto," in: Winograd, T.A. (ed.), Bringing Design To Software, ACM Press, 1996, pp. 1-9.

Kelly, G.A. The Psychology Of Personal Constructs., W.W. Norton & Company Inc., New York, NY, 1955.

Khushalani, A., Smith, R., and Howard, S. "What Happens When Designers Don't Play By The Rules: Towards a Model of Opportunistic Behaviour In Design," Australian Journal of Information Systems, 1994, 1:2, pp. 13-31.

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

Kling, R., and Scacchi, W. "The Web of Computing: Computer Technology as Social Organization," in: Yovits, M.C. (ed.), Advances In Computers, Academic Press, New York, 1982, pp. 1 - 90.

Kuhn, S. "Design For People At Work," in: Winograd, T.A. (ed.), Bringing Design To Software, ACM Press, 1996, pp. 263-279.

Land, F. "The Information Systems Domain," in: Galliers, R. (ed.), Information Systems Research, Blackwell Scientific, Oxford UK, 1992, pp. 6-13.

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

Lau, F., Doze, S., Vincent, D., Wilson, D., Noseworthy, T., Hayward, R., and Penn, A. "Patterns of improvisation for evidence-based practice in clinical settings," Information, Technology and People, 1999, 12:3, pp. 287-303.

Lave, J. "Situating Learning In Communities of Practice," in: Resnick, L.B., Levine, J.M. and Teasley, S.D. (eds.), Perspectives on Socially Shared Cognition, American Psychological Association, Washington DC, 1991.

Lawson, B. How Designers Think, (Second Edition ed.) The Architectural Press, London UK, 1990.

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

MacKenzie, D.A., and Wajcman, J. "Introductory Essay," in: Mackenzie, D.A. and Wajcman, J. (eds.), The Social Shaping Of Technology, Open University Press, Milton Keynes UK, 1999, pp. 3-27.

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

Malhotra, A., Thomas, J., Carroll, J., and Miller, L. "Cognitive Processes In Design," International Journal of Man-Machine Studies, 1980, 12, pp. 119-140.

Markus, M.L., and Bjorn-Andersen, N. "Power over users: its exercise by system professionals," Communications of the ACM June 1987, 1987, 30:6, pp. 498-504.

Markus, M.L., Majchrzak, A., and Gasser, L. "A Design Theory For Systems That Support Emergent Knowledge Processes," MIS Quarterly, 2002, 3:3, pp. 179-212.

Markus, M.L., and Robey, D. "Information Technology And Organizational Change: Causal Structure In Theory And Research," Management Science, 1988, 34:5, pp. 583-598.

Mathiassen, L., and Stage, J. "The Principle of Limited Reductionism In Software Design," Information Technology & People, 1992, 6:2, pp. 171-185.

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

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

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

Nelson, E. "Extreme Programming vs. Interaction Design," 2002, Available at http://www.fawcette.com/interviews/beck_cooper/default.asp, last accessed August 2003.

Norman, D.A. The Design of Everyday Things, (Also published as: The Psychology of Everyday Things 1988, Harper-Collins, New York ed.) Basic Books, Doubleday, New York, NY, 1990.

Orlikowski, W.J. "The Duality of Technology: Rethinking The Concept of Technology In Organizations," Organization Science, 1992, 3:3, pp. 398-427.

Orlikowski, W.J. "Improvising Organizational Transformation Over Time: A Situated Change Perspective," Information Systems Research, 1996, 7:1, pp. 63-92.

Orlikowski, W.J. "Using Technology and Constituting Structures: A Practice Lens For Studying Technology In Organizations," Organization Science, 2000, 11:4, pp. 404-428.

Orlikowski, W.J., and Hofman, D. "An Improvisational Model of Change Management: The Case of Groupware Technologies," Sloan Management Review, 1997, 38:2, pp. 11-22.

Orlikowski, W.J., and Iacono, C.S. "Research Commentary: Desperately Seeking The "IT" in IT Research - A Call To Theorizing The IT Artifact," Information Systems Research, 2001, 12:2, pp. 121-134.

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

Preston, A. "The problem in and of management information systems," Accounting Management and Information Technology, 1991, 11, pp. 43-69.

Rittel, H.W.J. "Second Generation Design Methods," Interview in: Design Methods Group 5th Anniversary Report, DMG Occasional Paper, 1, 1972, pp. 5-10. Reprinted in N. Cross (ed.), Developments in Design Methodology, J. Wiley & Sons, Chichester, 1984, pp. 317-327.

Rosenbrock, H.H. "Engineering As An Art," AI & Society, 1988, 2:4, pp. 315-320.

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

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

Simon, H.A. "The Structure of Ill-Structured Problems," Artificial Intelligence, 1973, 4, pp. 145-180.

Strauss, A.L. Continual Permutations of Action, Aldine de Gruyter, New York, 1983.

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

Turner, J.A. "Understanding The Elements Of Systems Design," in: R.J., B. and R.A., H. (eds.), Critical Issues In Information Systems Research, Wiley, New York, NY, 1987, pp. 97-111.

Weick, K.E. The Social Psychology of Organizing, Addison-Wesley, Reading MA, 1979.

Weick, K.E. Making Sense of the Organization, Blackwell Scientific, Malden MA, 2001.

Wenger, E. Communities of Practice - Learning Meaning and Identity, Cambridge University Press, Cambridge UK, 1998.

Wilkinson, B. The Shopfloor Politics Of New Technology Gower Press, London UK, 1983.

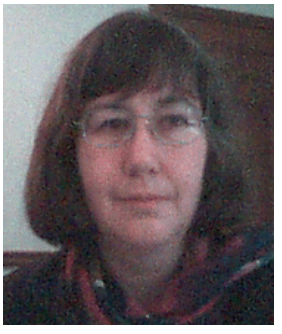
Winograd, T. "Introduction," in: Winograd, T. (ed.), Bringing Design To Software, ACM Press, Addison-Wesley Publishing, New York NY, 1996.

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

Xu, X.M., Lehaney, B., Clarke, S., and Yanqing, D.J. "Some UK and USA comparisons of executive information systems in practice and theory," Journal of End User Computing, 2003, 15:1, pp. 1-19.

Yates, J., and Orlikowski, W. "Genre Systems: Structuring Interaction through Communicative Norms," The Journal of Business Communication, 2002, 39:1, pp. 13-35.

AUTHOR



Susan Gasson (PhD, Warwick Business School, UK) is Assistant Professor in IS at the College of Information Science and Technology at Drexel University. Her research interests include social cognition, the design of organizational information systems and human-centeredness in information system implementation and use. Prior to her academic career, she worked as a consultant and manager in the areas of software design, data communications and information systems architectures.