December 2007

Information System Architecture: Toward a Distributed Cognition Perspective

David Dreyfus
Boston University

INFORMATION SYSTEM ARCHITECTURE: TOWARD A DISTRIBUTED COGNITION PERSPECTIVE

David Dreyfus
Boston University
595 Commonwealth Avenue, Boston, MA 02215
ddreyfus@bu.edu

Abstract

Organizations make substantial investments in their information system architectures, yet extant theory on information system architecture often regards the architect as having a great deal of design freedom. This paper argues that information system architecting can be profitably viewed as distributed cognition in which multiple decision-makers influence the evolution of an information system's architecture. A set of guidelines are constructed from the literature that can be used to characterize the degree to which a set of activities constitutes distributed cognition. Interview data is then used from three sites engaged in information system architecting to illustrate why architecting is difficult and how a distributed cognition perspective helps us understand it. The paper concludes by arguing that although the architecting process can be described as distributed cognition, the practice of architecting is constrained by the absence of representations that facilitate cross-group discussions of the architecture.

Keywords: Architecture, emergence, network, distributed cognition, boundary objects, qualitative study, interviews, information system architecture

Introduction

Organizations make large investments in IT-enabled information systems expecting them to produce multiple benefits; among these are faster and cheaper transmission, manipulation, analysis, and exploitation of information in order to improve and more optimally distribute organizational decision making, enhance environmental scanning, and share organizational expertise (Huber 1990). We know from prior research that how the components of the information system are interconnected – its architecture – affects the benefits it affords (Byrd et al. 2000; Duncan 1995; Henderson et al. 1993; Kayworth et al. 2001; McKay et al. 1989; Perry et al. 1992; Zachman 1987). We also know that how systems are developed and maintained is complicated by the way IS decision-makers and their stakeholders frame the issues around their design, deployment and maintenance (Boland et al. 1994; Jacko et al. 2003; Markus et al. 2002; Walls et al. 1992). Therefore, we want to develop a richer understanding of the architecting activity in order affect changes that may ultimately increase the benefits afforded by information systems.

Information systems have been viewed as systems of components interacting with each other in a coordinated fashion (Baldwin et al. 2000; Brooks 1975). An architecture team develops the system's architecture and component designers develop the individual components. Recognizing that there were multiple stakeholders involved in system design, Zachman (1987) developed a framework for designers working at different levels of abstraction serving different stakeholder requirements. At a finer level of granularity, researchers have looked to document design patterns that serve enterprise-level architecting (Schwinn et al. 2005). In each of these streams, design is the province of a single designer – an architect – or of a design team responsible for developing the information system architecture. As the scope of a system increases, however, design challenges become increasingly organizational.

The importance of organizational factors has been identified in large application development projects (Curtis et al. 1988; Markus 1983). However, integrated information systems are more than just a portfolio of large applications. The information system also includes the dependencies among the applications and supporting infrastructure elements. Missing from the information system design literature are the roles, assumptions, and goals of the multiple organizational actors responsible for creating, modifying, and maintaining the overall information system design. Individual decision-makers with organizationally defined roles, enacting organizationally defined routines, work in a coordinated fashion giving rise to what researchers reify as organizational decision making and learning (Grant 1996; Simon 1997 [1945]; Weick et al. 1993). The collective knowledge of organizational members is then preserved and carried forward in the artifacts and routines produced by the organization (Argote et al. 2003; Dawkins 1976; Huber 1990; Nelson et al. 1982). The central artifact in this paper is the overall information system; the routines that utilize and modify the artifact are distributed across different functional groups.

The purpose of this paper is to ground the information systems design process in its temporal and organizational context in which the design emerges as a result of multiple decision-makers modifying an evolving information system based upon their specific, local decision premises (Rogers et al. 1994; Simon 1997 [1945]). In order to do so, I build on the work of Boland, Tenkasi, and Te'eni who argue that decision making in organizations can often be properly recognized as distributed cognition, in which organizationally distributed decision-makers actively affect each others' understanding of the problems they are solving (Boland et al. 1994). Just as they anticipate a different set of tools and representations (artifacts) and routines that reflect the distributed cognition perspective to enhance communication among decision-makers, I also anticipate improved artifacts and routines that reflect a more accurate conceptualization of architecting to enhance communication and the sharing of architectural representations among decision-makers (Carlile 2002).

I argue that the information system architectures that exist in practice (which may vary considerably from their intended designs (Iyer et al. 2004)) are the result of application design, enhancement, and maintenance processes that span functional group boundaries (e.g., infrastructure, applications, business units). Individual decision-makers make local design decisions regarding infrastructure and application design, deployment, updating, upgrading, and decommissioning that have enterprise-wide ramifications. While there was a time when individual applications (e.g., ERP, HR, CRM) could be considered independently of each other, for many organizations that time has passed. Design processes that may be appropriate for independent applications may not be appropriate for systems of interdependent applications. Information system design processes in which decision-makers recognize the impact of the overall information system architecture on their local decisions, and vice versa, may lead to improved information system flexibility and alignment with business objectives (Byrd et al. 2000; Duncan 1995; Henderson et al. 1993; Kayworth et al. 2001; McKay et al. 1989; Perry et al. 1992; Zachman 1987).

To develop this conceptualization of information systems design as organizationally and temporally distributed, I interviewed decision-makers within the IT departments of a mutual fund services company, a pharmaceutical company, and a manufacturing and distribution company. In each of these companies organizationally and temporally distributed architectural decisions have constant reverberations, complicating other decision making processes. The following description offers a hint of the challenges information system architects face in practice.

> Adam[1] is an information systems architect responsible for supporting a pharmaceutical research group - scientists. He has responsibility for 60-70 applications – a subset of the applications within a research division. This responsibility includes adding new applications, retiring and replacing outdated applications, and maintaining and enhancing of existing applications. The infrastructure his applications run on (e.g., UNIX and Windows, database, and web application servers) are managed by a sister IT department.
>
> Adam reports, "I like to take myself as a bridge builder. I understand a lot of the science and a lot of the IT, obviously, and I have no problem spending time with people trying to explain things, how they work to them, but at the end of the day, they're really not that interested. They have a problem that they want solved and they come to us to solve the problem." The applications, IT, and scientific groups each have their own areas of concern, yet directly affect each other and the performance, reliability, functioning and evolution of the information system that binds them.

The conceptualization developed in this paper will help explain why managing the evolution of the information system is so difficult.

The remainder of the paper outlines a theory of how information system architectures evolve, develops the characteristics of a design space that indicates a distributed cognition problem, introduces the data, and shows how

[1] Not his real name. All names have been changed to provide anonymity to the informants.

the empirical data supports the distributed cognition perspective set forth earlier. The paper then concludes with a discussion and suggestions for further research.

Information system architecture

Architecture definition

There are many perspectives on what constitutes architecture (Iyer et al. 2004; Malone et al. 1994; Messerschmitt et al. 2003; Nezlek et al. 1999; Ross 2003; Zachman 1987). I characterize an organization's information system architecture as a state description of the software components, and the dependency relationships among them, deployed by the organization (Simon 1996 [1969]). A software component is defined as a unit of software that can be independently developed, installed, and executed (Hopkins 2000; Messerschmitt et al. 2003). Examples of software components include web application servers, database management systems, and ERP applications, each of which can be large, complex systems. Software components depend on each other for data and processing.

Architectures defined through a design process can be considered the espoused or normative view. However, whether designed or not, and whether recognized and documented or not, all organizations have an information system architecture. The architecture of the actual, deployed software components is the emergent architecture (Alexander 1964; Iyer et al. 2004) and is the focal artifact in this research. Although the espoused architecture may be unchanging, the emergent architecture evolves as the portfolio of deployed applications, and their interrelationships, change.

Architecture evolution

Architectures aren't just designed or implemented as a discrete activity; they evolve over time as individual decision-makers deploy, modify, integrate, and retire software components. These decisions are sometimes made by architects – organizational employees or consultants whose specific responsibility includes making these decisions. However, they are often made by other decision-makers who may or may not recognize the architectural implications of their decisions.

Common to all these decision-makers is that they operate with imperfect knowledge and understanding. They are limited in their knowledge of which software components exist outside of a particular decision-maker's sphere of decision making, the myriad ways components depend on each other, the interactions among the components, the technologies incident to the information technology, the business requirements, the user-requirements, and actual system use. They are ignorant of applications that will build upon or be retired from the existing system in the future. They are ignorant of each others' meanings and goals (Carlile 2002; Shannon 1948). Some of this ignorance is due to the volume of information, some is due to the stickiness of information (von Hippel 1994), and some of it is due to the combinatorial complexity of the potential interactions among the different sources of ignorance.

The limited, heterogeneous knowledge imperfectly shared by the decision-makers complicates coordinated action and results in architectures that may vary considerably from what any designer or design team envisioned, and from that architecture which would ideally suit the organization. Moreover, there may be no agreement on what the ideal architecture should be, or on what dimensions potential architectures should be evaluated (Feldman et al. 1981; Kling 1980; Markus 1983). Architectures that suit one part of the organization may be detrimental to another.

Changing the architecture is made more difficult by its path-dependent nature. As routines around the existing components are reinforced (Nelson et al. 1982; Orlikowski 1992), switching costs to new technologies (Farrell et al. 1986) increase. Not only do the users immediately affected by the change need to be trained, other changes to organizational structure and workflow may also be required to take full advantage of the changed information system (Brynjolfsson et al. 2002). Due to budget and time constraints that restrict an organization from making a wholesale replacement of its information system, organizations tend to make incremental change to their information systems instead. As a result, a firm's information system architecture tends to be unique.

Perspectives of architecting

To make sense of the activities that cause information system architectures to evolve, I first bound the activities that we will analyze and then sketch the distributed cognition perspective that may help us understand the activities more holistically.

Bounding the activity

This research focuses on the activities that change the information system architecture; therefore, included are those groups and individuals that directly influence the addition, removal, or integration of software components. Excluded are groups that indirectly influence the architecture (e.g., the HR department that hires the decision-makers that make architectural decisions). These other groups might indirectly set the limitations, conditions, political environment, and organizational culture in which architects and other decision-makers operate. As a result, these other groups might affect the information system architecture that emerges. However, this research focuses on the communication patterns and processes that directly change the architecture and not on the other factors that affect specific architectural outcomes.

Architecture as a design activity

Brooks (1975) divides system design activity into architecture and implementation. The architect is responsible for the conceptual integrity of the system, which he argues is the test of the design, and communication between the users and system developers. The architect is responsible for the interfaces the users see as well as the interfaces among the components that constitute the system.

The traditional view of design suggests that the architect start by collecting requirements from users and then creating a design that satisfies those users along a number of dimensions (Simon 1996 [1969]). The process of collecting those requirements may be quite involved, many stakeholders may be consulted, and different abstractions of the requirements may be presented (Zachman 1987); but, once this process has been completed, the architect can construct a solution based upon the best available technologies (Perry et al. 1992).

Kling (1980) described this traditional view, with its rational and structural perspectives, as systems rationalism. There are a number of important assumptions behind this approach. First, that the requirements for the system can be enumerated and agreed upon. Second, that the desired system can be modeled (Yourdon 1989; Zachman 1987), and that such modeling will make sense to the stakeholders. Third, that analysis of the desired system precedes construction of the system (Yourdon 1989). Fourth, that the proposed solution can be evaluated along technological dimensions (Kling 1980). Fifth, that knowledge about the problems can be separated from the solutions (Boland 2002; Simon 1996 [1969]).

Clearly, there are many examples of situations where the assumptions listed above are valid and the rational and structural perspectives produce terrific results. Such perspectives were reflected in the interview data I collected regarding application design and development. As the number of groups involved in system use increases, Kling (1980) suggests that a segmented-institutionalist approach may be more helpful than the systems rationalist approach. As the system changes in scale it may also change in type, requiring a different approach (Curtis et al. 1988; Markus 1983).

Due, in part, to the success of architects and other designers, individual systems (e.g., applications) have become increasingly integrated, resulting in larger, more complex, information systems. For the information system as a whole, there is generally no single architect or architecture group responsible for its design.

The distributed cognition perspective

From the American Heritage Dictionary, cognition is defined as “The mental process of knowing, including aspects such as awareness, perception, reasoning, and judgment.” To distill the distributed cognition perspective, I utilize the work of Hutchins (1995) and Boland, Tenkasi, and Te’eni (1994). Hutchins described an account of a navigational crisis on a navy ship. A key activity during the crises was pinpointing the exact location of the vessel. Maintaining the location of the ship as it moved around its anchor provided a clear, well understood, accepted goal; but the process by which the goal was achieved emerged over time. It was not designed a priori. A solution that combined the right combination of computational and social structure was found, and it was properly characterized as one that embodies distributed cognition. A slightly augmented set of characteristics for distributed cognition from Hutchins is:

1. The amount of computation required to perform the task exceeded the organizing, communicating, memorizing, recollection, and analyzing capabilities of a single person (Simon 1997 [1945]) which leads to a distribution of work and
2. a computational structure driven by the availability of data;
3. an organization of computation that is affected by the artifacts in use (Carlile 2002);
4. taking advantage of advantages of modularization and specialization;
5. and the necessity of a fit between the computational and social organization.

Individual actors minimize their work and organize their work around data as it is available from the environment. However, the final calculations depend upon the aggregate data provided by individuals via a social structure. Thus, the final description of how the task is performed is a joint function of computational and organizational structure.

Boland, Tenkasi, and Te'eni (1994) examine distributed cognition from the perspective of a non-directed search for understanding rather than a search for a specific answer to a specific problem. One of the central premises in that work is that the problem with too many systems is that the underlying conceptualization of decision-makers as computational engines that need data and processing power was incorrect for a great many problems. Instead, Boland, Tenkasi, and Te'eni characterize decision-makers within organizations as "interpreters and enactors of a stream of events in their organization." (p. 456)

In order to operate as interpreters and enactors, decision-makers "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." (p. 457) This independent yet interdependent decision making is not a shared cognition or a shared mind, but the recognition, appreciation, and respect of the abilities, needs, and perspective of the other (Nelson et al. 1996).

In Hutchin's conception of distributed cognition the goal the organization is trying to achieve is known, but the process of identifying and distributing the tasks to achieve the goal emerge and evolve through a process in which the individual decision-makers semi-independently come to understand their role and the tasks they must perform. In Boland, Tenkasi, and Te'eni's conception of distributed cognition both the goal and the means of identifying and distributing the resulting tasks emerge and evolve. Both of these conceptualizations differ from distributed problem solving in which both the goal and means for achieving it are known a priori.

While making no claim for completeness, the following minimal set of requirements would seem to need to be met before an organizational activity could be described as cognitively distributed.

1. The activity cannot be removed from its organizational context. The requisite data, knowledge, and artifacts are sticky (von Hippel 1994).
2. The understanding required to support an activity is beyond the capacity of a single individual.
3. Individual decisions can be made – a shared mind is not required (Weick et al. 1993).
4. Local decisions directly affect other decision-makers, creating interdependence among decision-makers.

The distributed cognition literature also emphasizes the importance of shared representations, organizational and computational fit, mutual taking each other into account, and engaging in meaningful conversation. I have excluded these characteristics from the definition of distributed cognition because they are primarily artifacts or activities that improve outcomes, or they are normative assessments of good process. They are not definitions of a distributed cognition situation. Shared representations, for example, may improve communication and help individuals take each other into account, but the absence of a shared representation doesn't mean that the situation isn't best described as distributed cognition. Similarly, meaningful conversation may improve decision enactment and lead to faster, more accurate problem solving, but its presence or absence isn't an indicator of distributed cognition.

Method

Prior research has ascribed certain benefits to information systems but hasn't provided detailed insight into the activities that lead to changes in information system architectures. I used interviews and document analysis to develop more detailed knowledge about why information system architecting is so difficult and how these organizations wrestle with issues related to the information system architecture within their organizational contexts. I developed an interview protocol (available from the author) following the approach suggested by Lofland, Snow, Anderson, and Lofland (2006), conducted and transcribed the interviews, and performed initial and focused coding (Glaser et al. 1967; Locke 2001) on the interview data.

Interviews were conducted to better understand architecting at the information system level. Initially, there was no guiding, explanatory theory. Interviewing and coding were conducted in parallel. As interviews were transcribed and coded, the categories subsequently described as challenges started emerging. The generalized relations among them also soon became apparent. Distributed cognition theory was then recognized as providing a conceptual explanation for the patterns in the data. The categories and generalized relations were solidified and became saturated as the interviewing progressed. The generalized relationships were confirmed in follow-up interviews with the informants.

The data utilized in the subsequent analysis is the joint work of the participants and the researcher as questions were refined, answered, and followed-up on during the interview process (Holstein et al. 2004). The ultimate validity test is to be able to see that the data provide a complete and compelling portrait of the activities described, the data seem internally consistent, and the explanation seems right. The roles and outlooks of the participants are identified in the interviews themselves.

The data were collected over a one year period (as part of a larger, ongoing research project regarding the emergent nature of information system architectures) involving one to three, open-ended, semi-structured interviews with each participant. Each interview lasted approximately two hours. The data also include project management, Power Point presentations, and ad-hoc notes and diagrams provided by the participants. Participants include managers and staff members from architecture and IT groups that interact with users, infrastructure providers, and other architects.

I explored the difficulty of architecting with a purposeful sample of 10 people across 3 organizations. Although the organizations were in different industries – biopharmaceutical, mutual fund servicing, and manufacturing – they were surprisingly consistent in their organizational structure around their information system, the state of their information systems, and the issues that they faced. In each organization there were three primary communities that interacted with the information system: an infrastructure group that managed computer hardware, networks, and software server acquisition, deployment, and support; an application group that managed requirement analysis, software application acquisition (purchase, deployment, or a combination of the two), software application deployment, and support; and the various IT consuming groups (i.e., manufacturing, scientists, line-of-business groups, etc.). In each organization the information system consisted of hundreds of interconnected application. And, in each organization decisions made by one group affected the performance and decision-making of the other groups.

BioPharm is a $500 million biopharmaceutical company with over 1,000 employees. They engage in an extensive R&D process. I interviewed five managers and architects within the informatics group supporting scientific research. Within the informatics group there are multiple architects responsible for between 50 and 70 applications each. The architects are responsible for making sure that the scientific community has the applications they need to support their research activities by managing application acquisition, development, and retirement. The architects must balance evolving scientific requirements, vendor upgrades, licensing arrangements, and changing infrastructure (e.g., networks, operating systems, web application servers, database servers, and file servers) while managing costs and system reliability. The architects must understand and adapt to the decisions made by the scientific users (their clients) and the infrastructure group that provides the enabling networks and hardware and software servers.

MutServ is an organization that provides processing services to the investment management industry. They globally provide full service transfer agency and accounting services and have over $1 trillion in assets under management. I spoke with the chief architect and two members of his staff. They utilize a variety of IT technologies including mainframe systems running batch programs written in COBOL, Visual Basic applications using client-server style programming, and N-tier applications utilizing web application servers.

MutServ has grown internally and through acquisitions. As companies were acquired, each line-of-business (LOB) maintained much of the decision rights to control most aspects of their (line-of) business, including design, sales, back-end processing and IT services. The resulting enterprise architecture has duplication of functionality, limited integration, and a wide variety of user interfaces. The information system evolved absent a central design or design team. Although they have a chief architect and reference architectures, decision making is still distributed among the central architecture group and the various LOB managers.

ManLog is a publicly traded company with sales in excess of $1 billion, over 8,000 employees, and over 80 facilities. I spoke with the IT director and his data center manager. Although their public documents describe their focus as design and manufacturing, discussions with their IT department suggests that the most important driver of IT spending is logistics. The IT organization consists of 60+ people including business analysts (not necessarily technical), a software group, a data center group, and a help-desk support group. The data center group is responsible for UNIX and Windows hardware acquisition and maintenance, networking components, PC desktop architecture, and the acquisition and maintenance of various database, file, and web application servers (infrastructure components). The software group develops in-house applications, modifies commercial software applications, and integrates applications as needed.

The interviews in this research provide a window on the activities that change information systems from the perspective of those whose principal responsibility is information system architecture. If such activity can be properly characterized as involving distributed cognition, then it should be evident through this window. Therefore, these interviews and documents provide a weak but naturalized experiment through which to understand the particular challenges facing information system architecting. The purpose of these interviews is not to test the theory developed, but to illustrate its value in describing information systems architecting.

Architecting in the field

The purpose of this section is to show why architecting at the information system level so difficult and to relate these difficulties to the theoretical elements of the distributed cognition perspective. For the informants in my cases, architecting is the process of providing an information system that meets the needs of the IT consuming groups (e.g., scientists, line of business users, etc.). The interviews surfaced eight overlapping challenges the informants face in their roles. These challenges and their relationships are summarized briefly below. The summary is followed by the supporting data from the case studies. In the subsequent section, the challenges are generalized into the four elements of the distributed cognition perspective.

1. **Scale**. Each organization has over a hundred applications in different lifecycle phases.
2. **Integrated applications**. Integration among applications increases dependencies among them, complicating the dependencies within and among organizational groups.
3. **Distributed decision making**. Decision making regarding the information system is distributed across multiple people in line-of-business, infrastructure, and application groups.
4. **Incomplete and distributed application knowledge**. There is no single repository (human or otherwise) containing knowledge of the purpose, functionality, or implementation detail of all the applications and their interdependencies.
5. **Local optimization with global ramifications**. Each group engages in local optimization, but because these optimizations affect the shared information system, each group is affected by other groups' decisions.
6. **Mismatched communication.** The line-of-business, infrastructure, and application groups utilize different languages and artifacts in their communication.
7. **The information system is cumulative.** New technologies and architectures (e.g., mainframe, client-server, N-tier) are added as old technologies and architecture remain. The information system cannot be replaced; it can only be modified over time. Architectural roadmaps are guides not blueprints.
8. **Limited central architecture authority.** Central architecture groups have influence, but not control, over decisions that affect the information system architecture.

Figure 1 diagrams the apparent relationships among the challenges. The scale of the information system leads to specialization within and among the application, infrastructure, and user communities. Specialization then leads to distributed decision making. The cumulative nature of the information system combines with its scale to lead to incomplete, distributed knowledge. Distributed decision making seems to lead to local optimization, but because the objects of those decisions are integrated, the ramifications of the decisions are more global. Distributed decision making seems associated with mismatches in communication as each group develops specialized languages. Distributed decision making coupled with incomplete knowledge seems associated with the limited writ of centralized architecture groups.

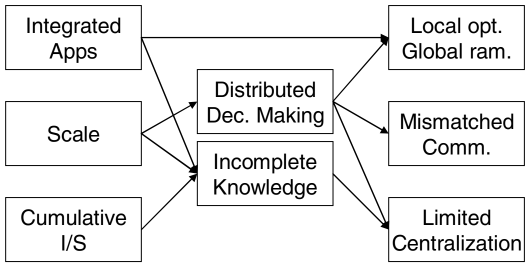

Figure 1: Influence relationships among challenges

Scale

The first challenge is scale. Each organization has over a hundred applications in different lifecycle phases. These applications are specialized for specific use. Applications are continuously being added, modified, and decommissioned. The information systems directly support thousands of users with different requirements. There are corresponding scale issues managing the supporting infrastructure.

At ManLog's corporate level there are 30 applications, one of which is ERP. There are about 15 ERP systems in the company. Each division also has multiple applications, and the factories have their own software systems. The ERP system consists of 11,000 modules. ManLog has modified 3,000 of them. Each application consists of database and application servers plus the business logic.

> Anything that rises to the level of an app has probably two machines that are clustered together for failover. And then a third machine that is going to be a development machine. And then a fourth machine that would be offsite someplace for disaster recovery. So an app has at least four machines that have to be monitored. (ManLog IT director)

As the architectural focus moves from the application level to the information system, the nature of the challenge doesn't just grow in scale – it changes in type.

> My prior work was probably focused on a particular application - coding a particular application. The role I've grown into here is owning a whole swath of 60 applications and their interdependencies. My job is to think about how to create better design, minimize maintenance, and minimize downtime. I think more about the issues around a larger environment. (BioPharm architect)

Making decisions about IT is distributed among multiple people in separate groups

The second challenge is the distributed nature of decision-making. This challenge seems to be a direct response to the challenge of scale. In order to manage the scale of the information system and the conflicting demands placed upon it, organizations decentralize decision-making to differentiated groups. The first division is between IT and the business (or scientific user). The builders and users of systems are frequently not the same people. This division complicates the process of understanding what the application requirements are. The second division is between application support and infrastructure support. Within these three groups – users groups, application groups, and infrastructure groups – there are additional subgroups.

The separation of concerns between the applications and infrastructure groups is highlighted at ManLog. The two groups refer to the same set of resources by different names, reflecting the different priorities they place on them. For example, the infrastructure group names each machine within a cluster, the applications group just refers to the cluster.

At BioPharm, as long as the applications group can utilize the existing infrastructure environment, they are free to deploy new applications without consulting the infrastructure group. Similarly, the infrastructure groups make autonomous decisions regarding server consolidation, server upgrades, and the decommissioning of support for certain platforms.

The following quote captures both the independence and interdependence of decision-making.

> So, we want to upgrade this application, they want to upgrade this OS. They want to consolidate this server environment. So we are both bouncing changes off of each other. (BioPharm architect)

Groups engage in local optimization with global ramifications

The third challenge is that each group's local decisions impact the other groups. The impact of one group on another is mediated by the information system, making the impact less obvious at the time the decision is made. For example, to protect the company against computer viruses, fix bugs, or add features, infrastructure assets are patched and upgraded on a regular basis. Sometimes, however, applications break, and in rare instances take down a production facility (example from ManLog).

The application support groups try to test applications on test systems prior to putting patches into production, but some problems only surface afterwards. Other times, applications that were installed outside the auspices of any IT group, and are thus not tested, are discovered only when they break and a call is then logged with the IT group (example from BioPharm).

Efficiency decisions made by one group can affect reliability within another. For example, at BioPharm the architectural decisions the infrastructure group makes (e.g., collocating multiple schemas in a single instance) affects information system reliability and the application group's work effort. Upgrading the single database instance forces a retesting effort of all applications, causing scheduling problems. If all database schemas weren't tightly coupled to a single database instance, upgrades and associated testing could be more incremental. In another BioPharm example, a recent database upgrade broke one application. Because of the tight coupling between all applications and a single instance, rolling back the upgrade was just as problematic as resolving the new issue.

Optimization strategies can be in conflict.

> I'm trying to reduce [application integration]. The scientists say that they want to see all this information in one space. They want to create a dashboard type of thing, but that creates a high degree of maintenance on the backend. While it's great for the scientists, it's more of a nightmare for us. There so many dependencies that you're creating to create a single view into all these different data stores. (BioPharm architect)

Applications are integrated

The fourth challenge is that applications are increasingly integrated, increasing the number of dependencies among them. The implication is that the number of ways that a change in one application can affect another has grown significantly.

> You also have to take into consideration that six years ago, when I started here, we had silos of applications. Nothing was connected. So, over the last six years, you can see our environment changing to where everything is connected now. And the mentality, even though it's been six years, I don't think it's really caught up. (ManLog IT director)

A recent ERP upgrade at one site took two years because:

> It is interconnected to everything: there were 30 integration points on the list of stuff we tested for the ERP system. One application is connected to 30 other apps. (ManLog IT director)

Whereas the third challenge highlighted the interdependency between groups, mediated by the information system, this fourth challenge highlights the complexity of the information system itself, complicating all aspects of application acquisition, deployment, upgrading, and retirement.

> So, we may have an application that is in a dependency web or something. We are trying to pull it out and replace it with a new one. We have to train the users on that as well as build a web of interdependencies on the new application as well as maintain the dependencies on the old application until we can pull it out. (BioPharm architect)

The decisions made in infrastructure groups affect the applications groups and vice versa. The decisions made by the applications groups affect the user communities and the user communities have a direct involvement in the creation of new applications.

> And now I have the problem domain of 70 pieces of software. Beyond that, [another architect] is right across the hall. He has 50 pieces of his own software and it's not like there's a clean dividing line, right. The chemistry guys certainly want to see some of the biology data. And the bio-heads want to see some of the chemistry data. (BioPharm architect)

Application knowledge is incomplete and distributed

The fifth challenge is that there is no single repository (human or otherwise) of the existing applications, or the integrations between them. Different individuals know about different applications, different integrations, and at varying levels of understanding. This is a direct consequence of issues of scale, distributed decision-making, and complexity.

Almost all the informants are working to improve their understanding of the application landscape so that the most common way of uncovering new dependencies among systems isn't unexpected system faults. The IT groups have no mechanism for easily collecting and maintaining the application dependency data. Developing and maintaining such an inventory would require the involvement of multiple groups and significant efforts. Understanding the dynamic interdependencies would be even more involved than creating an inventory. While they agree that having the data would certainly change the way they think about their decisions, they are not sure if the cost of collecting and maintaining the data would be worth the benefit.

Although architects at all three sites construct abstractions of the information system architecture, they are all incomplete. At no time can the architects construct an abstraction that will accurately represent either what is or will be. They neither have a complete application inventory nor do they know how the applications interconnect.

> The infrastructure folks don't seem to want to care about what we do in informatics. And most of the people in informatics don't seem to want to care what they do in infrastructure. It's like we have enough of our own problems and our own concerns every day in our own work that maybe we don't have time to do it, but it seems critical. (BioPharm architect)

> Your environment just organically grows. Almost. Without the knowledge of the whole suite of systems, it is very difficult to make informed decisions. So, you do the best you can with the knowledge you have. You don't necessarily make the correct decision in light of the entire environment. You are making, in some sense, uninformed decisions. (BioPharm architect)

The application managers at each company in the study had documents describing the design of individual applications, and software development tools for the creation of new applications. What they do not posses are tools to track and manage the "organic" growth of the information system as a whole.

Mismatched communication

The sixth challenge is mismatched communication among groups, and the lack of tools and artifacts that can enable improved communication. Although each group depends on each other, their levels of specialization have led to group specific languages that thwart effective communication. The applications groups don't fully understand the issues the infrastructure groups have. They don't understand the group-specific language and challenges. Similarly, the applications groups don't fully understand the users. A developer at BioPharm said that one of the key factors for a successful application architect is learning the users' language and explaining issues to them, and asking questions of them, in that language. According to an architect at BioPharm, the users have no interest in understanding the information system architecture; however, if they did, they would better understand reliability issues and the reasons some features are more difficult to provide than others.

Politics, separate areas of concerns, and blame shifting seem to characterize inter-group relations due to the lack of shared interests (Nelson et al. 1996). Two types of activities between the user and application groups seem to mitigate mismatched communication. First, the application group will sometimes embed staffers within the user groups in order to develop a better understanding of requirements. Second, the application and user groups engage in iterative design. Both of these tactics were employed at BioPharm and MutServ.

Iterative design may point the way towards improved communication. The most valuable communication tool between the application group and the user groups are the prototypes and applications developed through iterative design activities. According to a BioPharm architect, iterative design surfaces issues, enables communication, and fosters a collaborative rather than legalistic or confrontational environment. The application under development functions well as a boundary object since both parties can use it to learn from and communicate with each other (Carlile 2002).

> I find that the formal Software Development Life Cycle method does not work well. I prefer the much more iterative rapid development cycle. There are several problems with the first approach. If you spend a year gathering requirements, the technology is already changed. The other thing is that scientists don't often understand what you're asking them when you're asking software or IT questions. You work with them for a while and they'll give you an answer, though they really don't understand the impact of an answer, even though it's written down. And they've signed off on it. (BioPharm architect)

The applications and infrastructure groups share no common artifact that represents the entire information system. Although they share a common interface – the infrastructure interface – they have no physical artifacts or abstractions that encapsulate their shared interests. The actual artifacts they have in common – the information system – is too big and complex to function as an effective boundary object (Carlile 2002). Instead, they struggle to communicate shared interests through personal relationships, PowerPoint slides, and extensive use of whiteboards (examples from BioPharm). Unfortunately, these have proven to be inadequate for developing or sharing an understanding of the information system architecture.

The informants in this study are clearly aware of the communication issues.

> The business people who have business ideas of what they need to do cannot communicate with the IT people who have to support and build the applications. So, there's a terminology gap, there's a language gap. Some business people here come to you expressing what they want in terms of your technology. Some of them are incredibly detailed. I want you to marry these two and do this, this, and this. That's a problem because, like I said before, business people should ask for what they really need, not for how to do it. So, there is a BIG conflict on that. That may have come from before where the only language they could adopt was the IT language. Because the IT people come back and tell the business people in IT terms. And the business people look at them and respond, "I don't know what you are talking about. What do you mean an XYZ widget." So, that's a huge problem here. Infrastructure people, it doesn't matter, whatever the groups are, the languages are all different. They have no common way of sharing that. Which is why bridging that gap is so incredibly important - very important. (MutServ architect)

The information system is cumulative

The seventh challenge is that the information system is cumulative. This challenge thoroughly complicates the creation of a master architectural plan. New technologies and architectures evolve and are added to the information system faster than old ones are retired. People that know the technologies and applications are replaced more quickly than the applications themselves.

> Even if you had the funding to replace everything, you wouldn't. [You end up with architectures that are] very idiosyncratic to domains within the organization. (BioPharm IT manager)

At MutServ, the information systems contain a variety of IT technologies including mainframe systems running batch programs written in COBOL, Visual Basic applications using client-server style programming, and N-tier applications utilizing web application servers. Due to the way organizations grow, and decision rights are allocated, the resulting enterprise architecture has duplication of functionality, limited integration, and a wide variety of user interfaces.

Architects within the organizations I talked to create roadmaps that they can use to guide their information systems, but they can’t replace their existing information systems and implement a new architecture.

> And, again, the Enterprise Service Bus model is a theoretical goal. It's not fully implemented. That picture there is an end-game - a road-map. When we look at stuff, we are trying to put it into that architecture, but not in all cases has it been done. It hasn't been done in most cases. (ManLog IT director)

No central architecture authority

The eighth challenge is that even when there is a chief architect role, the power and knowledge of that architect is limited, in part because of the first seven challenges. At MutServ the chief architect can create architectural review boards and define road-maps, but the architect's writ is ultimately limited to influencing the distributed decision-making described earlier. Business managers with immediate objectives can bypass the review boards or fail to make the extra investment sometimes required for good architecture. The architect is unable to replace the existing information system (the seventh challenge) to implement a clean architecture, and doesn't have the knowledge required to do so anyway.

As the chief architected of MutServ pointed out, there are no diagrams or manuals that explain the services the company provides to its customers, or how those services should be provided. The requisite knowledge regarding what should be built is accumulated through meetings and discussions. The architecture must incorporate the process logic locked in the existing systems, yet there is no inventory of those systems, or what and how they communicate. Knowledge of what the systems do and what they need to do is distributed across organizational actors and artifacts that have yet to be fully discovered.

Central control over architecture seems limited by issues of scale, the difficulty in sharing information, distributed authority, and the absence of artifacts.

> I won't say we suffer from a lack of strong architecture, I won't say we suffer from a lack of strong engineering, or an engineering plan, or a structured model around our architecture. What we suffer from is artifacts associated with the use of the information that architecture model would or could or should generate. So, when we need write a test plan, when we need to install a change, a patch, whatever, in a test environment, what do we test to ensure that the change that we made doesn't break anything? So, that's the absence of a positive effect. (BioPharm IT director)

Interpretation

In the previous section I showed eight challenges associated with managing an information system's architecture. In this section the challenges are generalized as representing the four elements of distributed cognition described earlier. The results are summarized in Table 1 and detailed below.

Table 1: Relationships among challenges and the four elements of distributed cognition

	Contextual inseparability	Computational complexity	Separable decisions	Interdependent decisions
Scale		X		
Integrated applications		X		X
Distributed decision making	X		X	
Incomplete and distributed app. knowledge	X		X	X
Local optimization/ global ramifications	X			X
Mismatched communication		X		
Cumulative information system	X	X		
Limited central architecture authority			X	X

Element one: activity inseparable from context

The first element of the distributed cognition perspective is that the decision-making that affects the information system architecture cannot be separated from its organizational context. In this study, four challenges seem to anchor architectural decision making to the specific organizational context: distributed decision making, incomplete and distributed application knowledge, groups engaging in local optimization with global ramifications, and the cumulative nature of the information system.

The information and expertise necessary for decision making that impacts the information system architecture at each study site is distributed among multiple groups. The groups, however, are forced to share a certain amount of information, and adjust to the decisions made by other groups, because decisions made by one group affect the other groups. The incomplete and specialized knowledge regarding the applications, infrastructure, user requirements, and their interdependencies makes the information sticky (von Hippel 1994). The stickiness of the information makes separation of architectural decision making from the individuals making the decisions difficult (Carlile et al. 2003). The cumulative nature of the information system makes it more idiosyncratic to the organization. The specific selection of applications that constitute the information system, and how those applications interconnect, reflects the specific needs and history of the groups interconnected through the information system (Briers et al. 2001). The result is that architectural decision making is inseparable from its organizational context.

Element two: the understanding required is beyond a single individual's ability

The second element of distributed cognition is that the activity is beyond the cognitive ability of a single individual or group. In this study, four challenges seem to make management of the information system architecture more computationally complex than a single person or group can handle: scale, integrated applications, mismatched communication, and the cumulative nature of the information system.

The number of applications, infrastructure elements, and users increases the information processing requirements beyond the capacity of a single person. However, dividing the information processing among multiple people or organizational groups only partially solves the information processing challenge because the applications, infrastructure elements, and user requirements are integrated. This integration results in interdependencies among the various decisions. These interdependencies increase the information processing required above that which would be required to manage the individual applications, infrastructure elements, and user requirements in isolation. The specialization of decision making among the different groups leads to specialization of language (Boland et al. 1995) and subsequent mismatches in communication. The communication mismatches makes information and knowledge sharing more difficult – more computationally expensive (von Hippel 1994). The cumulative nature of the information system further increases the cognitive requirements by increasing the depth (historical) and breadth (number of applications) of information potentially incorporated into the decision making activities that affect the information system.

Element three: individual decisions can be enacted

The third characteristic of distributed cognition is that individual decisions can be enacted without the immediate regard of other decision-makers. Whereas the elements of distributed cognition previously described are concluded from the architectural challenges, the challenges in this section reflect the relative separability of decision making. In this study, three challenges reflect the observation that independent decisions can be enacted: decision making is distributed, application knowledge is incomplete and distributed, and centralized authorities have limited influence.

How the individuals in each department utilize the applications does not dictate how the applications are built or upon which pieces of infrastructure the applications reside. The formation of distinct groups within each company to manage different aspects information system growth, maintenance, and use suggests that many decisions made within one group can be made independently of the decisions made by other groups. The distribution of knowledge across groups reflects group-specific specialization and information processing requirements (Lawrence et al. 1967). The combination of distributed decision making and incomplete and distributed application knowledge seems to lead to a limited role for a centralized architectural authority.

Element four: individual decisions are interdependent

The fourth characteristic is that individual decisions are interdependent. In this study, four challenges seem to create interdependency among the decisions that affect the information system architecture: integration among applications, incomplete and distributed application knowledge, local optimizations with global ramifications, and limited centralized architectural authority. The interdependency among decisions can be described as loose coupling

(Weick 1976). Some decisions in each group do not affect other groups, and some decisions do. Some decisions have a large impact on other groups, and some do not. Loose coupling occurs both across time – decisions made in the past are coupled to those being made today through the existence of legacy applications – and across groups.

The integration among applications connects some decisions made to one application to some decisions made on another application. New user requirements, application updates and patches, and application replacement of one application can force corresponding changes to the applications it is integrated with. Incomplete and distributed knowledge means that the consequences of a change to an application are not always known. Because the applications are integrated, the consequences of a change to one application on other applications are also not always known. Similarly, changes in applications can influence changes in how the applications are used and changes in infrastructure can affect application reliability and performance.

The global ramifications of some decisions are a direct reflection of the interdependency among decisions. So, too, is the existence of limited authority exercised by centralized groups. In each of the cases, the more centralized IT functions arose to better manage the interdependencies among groups. At ManLog the central function provided corporate wide services (e.g., single sign-on and ERP applications), at MutServ the central architectural function tries to guide the lines of business towards good architectural choices, and at BioPharm the central architectural functions managed the competing, interdependent challenges faced by scientists and IT professionals.

Implications

This paper has characterized information system architecture as a description of the hardware and software components of an information system and their interdependencies. While the individual components of the information system, and specific integrations among them, may be designed, the overall information system emerges as a result of interdependent decisions enacted over time. The architecture is *guided* not *created*.

Large-scale information system architecting, at least as described in the three case studies presented in this paper, can be viewed as distributed cognition in practice. As architects move from concern over applications (even those that are built of multiple, interconnected components) to concern over how all of the applications interconnect at the enterprise level, the problem changes in terms of scale and type. There are multiple decision-makers involved in the evolution of the information system. Decision-makers from the distinct groups - users, infrastructure providers, and architects - have distinct world-views and interpretations of the information system. They are clearly interdependent on each other and seem to adjust their own world-views as a result of changes made by other groups via the information system.

The distributed cognition perspective can help us identify classes of problems that can be addressed through the perspective's normative implications. Identifying a problem does not imply that it is well solved. Characterizing architecting as a distributed cognition activity leads directly to the question of what are the routines and artifacts that would enhance the distributed architecting activities, and thereby gain greater insight into how distributed cognition can be proactively supported. Alternatively, research can explore how information systems can be shaped to reduce the need for distributed cognition.

The data supports the observation that a critical missing factor in the case studies is artifacts that can provide shared representations of the information system architecture (Boland et al. 1994; Hutchins 1995). These representations should have the capacity to function as boundary objects (Carlile 2002), enabling the groups using them to develop a more common language and a shared appreciation of the needs and perspectives of each other (Levina 2005; Nelson et al. 1996). By operating at multiple levels of granularity the representations facilitate understanding detail within a larger context and understanding the larger context in terms of finer levels of detail.

Architectural representations, modeling, and simulation tools are frequently used at the application level where design precedes implementation, and they perform well within that specific work context. How well do these existing tools and artifacts function at different levels of granularity? They didn't function well at the information system level in the case studies. At the overall information system level, tools that discover and aggregate individual application architectures don't seem to exist. Can existing tools scale up, or do a new classes of tools need to be developed?

In many organizations accounting systems provide an abstraction that different groups can manage to, and which function as boundary objects (Briers et al. 2001). Accounting systems reflect underlying activity but are not part of that activity. For example, the accounting system may reflect the production of widgets or the performance of mutual fund servicing, but it doesn't perform a strictly necessary step in the production or service process. However, accounting systems have been widely accepted as a way of measuring and communicating performance and have often been accepted as a representational mechanism among disparate groups. Similarly, are there representations of architecture that can become important to multiple groups and function as a measurement tool?

Although there has been some significant work on the importance of boundary objects that mediate interpersonal and inter-group communication, and some work on the characteristics of those objects, there has been no empirical testing of the characteristics of higher and lower performing boundary objects within the IT context. Measurement, monitoring, and control systems, however, may also function as boundary objects and yet do not necessarily mediate interpersonal and inter-group communication. In what contexts do these types of systems function well as a boundary object? What features or characteristics make enable higher or lower performance?

If one wanted to provide IT support for the creation and sharing of architectural representations, what should the features and capabilities of that system look like? It seems to me, based on discussions with my informants, that such a tool would have the following capabilities (there are, of course, many other possibilities).

1. Capture as automatically as possible a real-time description of the current architecture.
2. Enable different users to represent the architecture numerically, textually, and visually and different levels of abstraction.
3. Experiment with different representations and scenarios through drawing and simulations.
4. Map different architectures (e.g., infrastructure, deployment, application, and business) to each other to facilitate sharing across group boundaries.
5. Superimpose other measures such as problem reports, investments, clients served over architectural renderings to provide executive and operational dashboards.

The heart of the problem is managing complexity by identifying and understanding dependencies among an evolving set of information system components in such a way that different groups can make independent decisions that effectively take into account the needs and perspectives of other such distributed decision-makers. The key to the solution may be in the development of artifacts that are both at a manageable level of abstraction and are important to multiple groups.

References

Alexander, C. *Notes on the synthesis of form* Harvard University Press, Cambridge, MA, 1964.

Argote, L., McEvily, B., and Reagans, R. "Managing knowledge in organizations: An integrative framework and review of emerging themes," *Management Science* (49:4), 2003, pp 571-582.

Baldwin, C.Y., and Clark, K.B. *Design Rules: The Power of Modularity* MIT Press, Cambridge, MA, 2000.

Boland, R.J. "Design in the Punctuation of Management Action," in: *Managing as Designing: Creating a Vocabulary for Management Education and Research,* R. Boland (ed.), Frontiers of Management Workshop, Weatherhead School of Management, June 14-15, (http://design.cwru.edu), 2002.

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

Boland, R.J.J., Tenkasi, R.V., and Te'eni, D. "Designing information technology to support distributed cognition," *Organization Science* (5:3), 1994, pp 456-475.

Briers, M., and Chua, W.F. "The role of actor-networks and boundary objects in management accounting change: a field study of an implementation of activity-based costing," *Accounting, Organizations and Society* (26:3), 2001, pp 237-269.

Brooks, F.P. *The mythical man-month: essays on software engineering* Addison-Wesley Pub. Co., Reading, Mass., 1975.

Brynjolfsson, E., Hitt, L.M., Yang, S., Baily, M.N., and Hall, R.E. "Intangible assets: Computers and organizational capital / Comments and discussion," *Brookings Papers on Economic Activity*:1), 2002, pp 137-198.

Byrd, T.A., and Turner, D.E. "Measuring the flexibility of information technology infrastructure: Exploratory analysis of a construct," *Journal of Management Information Systems* (17:1), 2000, pp 167-208.

Carlile, P.R. "A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development," *Organization Science* (13:4), 2002, pp 442-455.

Carlile, P.R., and Rebentisch, E.S. "Into the black box: The knowledge transformation cycle," *Management Science* (49:9), 2003, pp 1180-1195.

Curtis, B., Krasner, H., and Iscoe, N. "A Field-Study of the Software-Design Process for Large Systems," *Communications of the ACM* (31:11), 1988, pp 1268-1287.

Dawkins, R. *The selfish gene* Oxford University Press, New York, 1976.

Duncan, N.B. "Capturing flexibility of information technology infrastructure: A study of resource characteristics and their measure," in: *Journal of Management Information Systems*, M.E. Sharpe Inc., 1995, pp. 37-58.

Farrell, J., and Saloner, G. "Installed Base and Compatibility: Innovation, Product Preannouncements, and Predation," *The American Economic Review* (76:5), 1986, pp 940-955.

Feldman, M.S., and March, J.G. "Information in Organizations as Signal and Symbol," *Administrative Science Quarterly* (26:2), 1981, pp 171-186.

Glaser, B.G., and Strauss, A.L. *The discovery of grounded theory; strategies for qualitative research* Aldine Pub. Co., Chicago, IL, 1967.

Grant, R.M. "Toward a knowledge-based theory of the firm," *Strategic Management Journal* (17:Winter), 1996, pp 109-123.

Henderson, J.C., and Venkatraman, N. "Strategic alignment: Leveraging information technology for transforming organizations," *IBM Systems Journal* (32:1), 1993, pp 4-16.

Holstein, J.A., and Gubrium, J.F. "The active interview," in: *Qualitative Research: Theory, Method and Practice,* D. Silverman (ed.), 2004, pp. 140 – 161.

Hopkins, J. "Component Primer," *Communications of the ACM* (43:10), 2000, pp 27-30.

Huber, G.P. "A Theory of the Effects of Advanced Information Technologies on Organizational Design, Intelligence, and Decision Making," *Academy of Management Review* (15:1), 1990, pp 47-71.

Hutchins, E. *Cognition in the wild* MIT Press, Cambridge, Mass., 1995.

Iyer, B., and Gottlieb, R.M. "The Four-Domain Architecture: An approach to support enterprise architecture design.," in: *IBM Systems Journal*, IBM Corporation/IBM Journals, 2004, pp. 587-597.

Jacko, J.A., and Sears, A. *The human-computer interaction handbook: fundamentals, evolving technologies, and emerging applications* Lawrence Erlbaum Associates, Mahwah, N.J., 2003.

Kayworth, T.R., Chatterjee, D., and Sambamurthy, V. "Theoretical Justification for IT Infrastructure Investments," *Information Resources Management Journal* (14:3), 2001, pp 5-14.

Kling, R. "Social Analyses of Computing: Theoretical Perspectives in Recent Empirical Research," *ACM Computing Surveys* (12:1), 1980, pp 61-110.

Lawrence, P.R., and Lorsch, J.W. *Organization and Environment; Managing Differentiation and Integration* Division of Research Graduate School of Business Administration Harvard University, Boston, 1967.

Levina, N. "Collaborating on Multiparty Information Systems Development Projects: A Collective Reflection-in-Action View," *Information Systems Research* (16:2), 2005, pp 109-130.

Locke, K. *Grounded theory in management research* Sage Publications, London; Thousand Oaks, Calif., 2001.

Lofland, J., Snow, D., Anderson, L., and Lofland, L.H. *Analyzing social settings: a guide to qualitative observation and analysis*, (4th ed.) Wadsworth/Thomson Learning, Belmont, CA, 2006.

Malone, T.W., and Crowston, K. "The Interdisciplinary Study of Coordination," *ACM Computing Surveys* (26:1), 1994, pp 87-119.

Markus, M.L. "Power, Politics, and MIS Implementation," *Communications of the ACM* (26:6), 1983, pp 430-444.

Markus, M.L., Majchrzak, A., and Gasser, L. "A design theory for systems that support emergent knowledge processes," *MIS Quarterly* (26:3), 2002, pp 179-212.

McKay, D., and Brockway, D. "Building IT Infrastructure for the 1990s," Nolan (ed.), Norton & Company, New York, 1989, pp. 1-11.

Messerschmitt, D.G., and Szyperski, C. *Software ecosystem: understanding an indispensable technology and industry* MIT Press, Cambridge, Mass., 2003.

Nelson, K.M., and Cooprider, J.G. "The contribution of shared knowledge to IS group performance," *MIS Quarterly* (20:4), 1996, pp 409-430.

Nelson, R., and Winter, S. *An Evolutionary Theory of Economic Change* Harvard University Press, Cambridge, MA, 1982.

Nezlek, G.S., Jain, H.K., and Nazareth, D.L. "An integrated approach to enterprise computing architectures," *Communications of the ACM* (42:11), 1999, pp 82-90.

Orlikowski, W.J. "The Duality of Technology: Rethinking the Concept of Technology in Organizations," *Organization Science* (3:3), 1992, pp 398-427.

Perry, D.E., and Wolf, A.L. "Foundations for the Study of Software Architecture," *ACM SIGSOFT Software Engineering Notes* (17:4), 1992, pp 50-52.

Rogers, Y., and Ellis, J. "Distributed cognition: an alternative framework for analysing and explaining collaborative working," *Journal of Information Technology* (9:2), 1994, pp 119-128.

Ross, J.W. "Creating a Strategic IT Architecture Competency: Learning in Stages," *MIT Sloan Working Paper No. 4314-03* (Center for Information Systems Research Working Paper No. 335), 2003.

Schwinn, A., and Schelp, J. "Design patterns for data integration," *Journal of Enterprise Information Management* (18:4), 2005, pp 471-483.

Shannon, C.E. "A Mathematical Theory of Communication," *The Bell System Technical Journal* (27), 1948, pp 379-423.

Simon, H.A. *The Sciences of the Artificial*, (Third ed.) The MIT Press, Cambridge, MA, 1996 [1969].

Simon, H.A. *Administrative behavior: a study of decision-making processes in administrative organizations*, (4th ed.) Free Press, New York, 1997 [1945].

von Hippel, E. ""Sticky Information" and the locus of problem solving: Implications for Innovation," *Management Science* (40:4), 1994, pp 429-439.

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

Weick, K.E. "Educational Organizations as Loosely Coupled Systems," *Administrative Science Quarterly* (21:1), 1976, pp 1-19.

Weick, K.E., and Roberts, K.H. "Collective mind in organizations: Heedful interrelating on Flight Decks," *Administrative Science Quarterly* (38:3), 1993, pp 357-382.

Yourdon, E. *Modern structured analysis* Yourdon Press, Englewood Cliffs, N.J., 1989.

Zachman, J.A. "A framework for information systems architecture," *IBM Systems Journal* (26:3), 1987, pp 276-293.