Process Coordination Requirements: Implications for the Design of Knowledge Management Systems

Bala Iyer[1]
Department of Information Systems, Boston University School of Management
595 Commonwealth Avenue, Boston, MA 02215
Phone: 617 353 4402; Fax: 617 353 5003
Email: bala@acs.bu.edu

G. Shankaranarayanan[*]
Department of Information Systems, Boston University School of Management
595 Commonwealth Avenue, Boston, MA 02215
Phone: 617 353 4605; Fax: 617 353 5003
Email: gshankar@acs.bu.edu

George Wyner
Department of Information Systems, Boston University School of Management
595 Commonwealth Avenue, Boston, MA 02215
Phone: 617 353 4153; Fax: 617 353 5003
Email: gwyner@acs.bu.edu

Revised on May 11 2006

[1] Author names in alphabetic order
[*] Corresponding author

Process Coordination Requirements: Implications for the Design of Knowledge Management Systems

Knowledge Management Systems (KMS) are typically built with a departmental focus, making it difficult to share and utilize knowledge across departmental boundaries. Integrating such "knowledge pockets" into a knowledge network requires resolving structural differences and coordinating both knowledge processes and software components. Here we identify a set of coordination requirements for the design of a KMS to support such knowledge networks .We first offer a classification of KM practices to define the types of KM methods and the system requirements for each. We then use coordination theory, specifically, Text-based Process Analysis, to analyze four texts (cases) that capture KM practices representing the different KM methods, to understand and identify the coordination requirements. We also propose the Enterprise Knowledge Architecture (EKA) as a standard for incorporating these requirements. The EKA supports building new KMS and analyzing existing ones by providing a common framework for designing, developing, and maintaining KMS. It thus provides a foundation for integrating knowledge pockets into knowledge networks.

KEY WORDS: Information Systems, Knowledge Management Systems (KMS), Knowledge Management (KM), Architecture, Coordination Theory, and Knowledge Networks.

1. INTRODUCTION

Knowledge management (KM) [1, 14, 24, 25, 33] has moved from an esoteric topic confined to academic institutions to a mainstay in organizations. KM applications are typically built within individual business units or departments and are focused internally [33]. This makes it difficult to share and utilize the knowledge contained in a Knowledge Management System (KMS) across departmental boundaries, resulting in isolated *knowledge pockets*. Each KMS is of limited value to outsiders because knowledge within that KMS must be interpreted using the thought models and vocabulary of its intended users. A KMS provides increased value when knowledge is shared across these boundaries and combined in new contexts.

Established systems analysis and design techniques such as use cases and JAD have demonstrated their utility in surfacing functional requirements to support user activities like adding entries into a KMS or searching a KMS for relevant information. However, sharing and integrating knowledge across departmental boundaries pose unique challenges - they require functionality to coordinate the interaction among multiple knowledge management activities as

well as support these activities individually. Systems design techniques do not typically address such process coordination requirements, relying instead on analysts to intuit these key functional requirements from complex process models such as flow charts and activity diagrams. These represent the current flow of activities *but not* the underlying dependencies among the processes.

Process coordination should be distinguished from the coordination of software components across an enterprise network. This *component coordination* has been addressed in the literature on Enterprise Application Integration (EAI) [4, 6, 17, 26]. However, the literature on EAI has not addressed the issue of coordinating KM processes and process resources. We believe that *process coordination* is an important ingredient for a knowledge network. Processes are the higher order construct involved in performing work that adds value to the enterprise. For example, in customer relationship management, to serve a customer, different sub-processes must be coordinated such that the customer gets a single view of the enterprise. One potential source of insight for understanding process coordination is coordination theory. It proposes a set of diagrams and methods for representing and analyzing coordination issues [7]. While researchers have evaluated its role in process innovation and internal coordination of software components [7], it is yet to be integrated with existing systems analysis and design approaches.

In this paper we explore how Text-based Process Analysis [7], an analysis method based on coordination theory, can be used to surface coordination requirements for knowledge management and how such requirements can serve as the basis for an architecture to guide the design of KMS. The objective of identifying these coordination requirements is to enable an organization to integrate a collection of knowledge pockets into a *knowledge network*. In other words, this study investigates the **research question**: *Does coordination analysis lead to new functional requirements for knowledge management and KMS?*

More specifically, this study evaluates two **research propositions**: ***(1) Coordination theory can identify the dependencies within the KM process and define coordination requirements for managing them. (2) Coordination theory can identify the dependencies among multiple knowledge processes and define the coordination requirements for managing the knowledge network.*** To understand the coordination requirements of a KM process, we will use coordination theory and TBPA to analyze several well-known KM alternatives. The result of our analysis is a high-level taxonomy of KM processes. Proposition 1 is demonstrated by identifying the taxonomy and the individual process maps using coordination theory. Proposition 2 is demonstrated by deriving a preliminary set of requirements for knowledge networks.

Although our focus in this paper is on KMS, the techniques proposed are applicable to the design of any system that captures, organizes, and disseminates data and which needs to be integrated with other systems to provide an organization-wide view of data. It follows that our approach is potentially applicable to the design of most business information systems (including, for example, transaction processing systems). It is also important to note that the methods of analysis used in this research complement (not replace) exiting systems analysis techniques. The requirements for individual activities in the knowledge management process constitute the *production requirements* for a KMS - requirements for carrying out the actual work of knowledge management. Existing systems analysis methods offer a number of effective techniques for surfacing such production requirements. This paper explores how to develop the requirements for coordinating these activities. By adding this additional analysis to existing methods, an analyst can surface additional *coordination requirements* and more effectively address the design challenges posed by coordination intensive systems such as KMS.

In section 2, we examine the KM literature to classify the approaches for KM and understand the requirements for each. In section 3 we use Text-based Process Analysis to develop a generalized process model for knowledge management based on several examples of the KM process. In section 4 we analyze this process model to develop a set of coordination requirements for both knowledge pockets and knowledge networks. In section 5 we use these requirements to construct the broad outlines of an Enterprise Knowledge Architecture (EKA) and give a brief example of how the EKA might be used to design a KMS. In section 6 we present some insights from this research and suggest directions for future work.

2. APPROACHES TO KM

We define KM as an organizational capability that allows people in organizations to create, capture, share, and leverage their collective knowledge to improve performance [2, 11, 33]. Knowledge may be tacit or explicit [24]. Hansen et al. [14] have identified the organizational approaches for the transfer of tacit and explicit knowledge as *personalization* (where computers help people communicate knowledge) and *codification* (where knowledge is captured in databases and accessed by others) respectively. By analyzing the KM literature from these two inter-related perspectives we identify two well-defined approaches for managing knowledge in organizations: (1) Shared experiences approach, and (2) Modular approach.

The Shared Experiences Approach: The underlying principle here is that knowledge is socially constructed, embedded in social networks, and it is the interactions between individuals and across organizations that create knowledge [5]. Central to this approach is the notion of communities, specifically, communities of practice (CoP) - groups of individuals both inside and outside of the organization who have a common interest and/or goal and who shape each other's behavior [29]. According to Nonaka, the knowledge creation process is not sequential, but a

spiral of interactions between tacit and explicit knowledge consisting of socialization, externalization, combination, and internalization [24]. Although explicit knowledge artifacts may be produced, capturing and reusing them is not the focus. The focus is on the just-in-time creation of knowledge by building on shared experiences such as the use of feed-back postings in e-commerce cites [22, 23]. Knowledge may not necessarily be captured and may be lost.

The shared experiences approach takes the view that (1) knowledge is difficult to reuse, and (2) the process of creating knowledge in an organization is difficult to control. This is because, in this approach, the community serves as a vehicle for knowledge creation, maintenance, and adoption, while organizational rules and policies simply support the process.

The Modular Approach: The economics of reuse has fostered this distinct approach for KM. Reuse presupposes codification, which is the process of converting organizational and individual tacit knowledge into explicit knowledge, in a standard form that makes it easily accessible, thus leveraging internal expertise on an enterprise-wide basis. The difficulty is to codify knowledge without losing its distinctive properties and specificity of context. Davenport and Prusak [7, 8], state that one must never lose sight of what business goals the codified knowledge will serve. They argue that organizations often provide either too little or too much structure to knowledge, hence decomposing it to data. Venkatraman and Henderson [28], in supporting *strategic intent* (as defined in [13]), concur with this principle when they highlight the importance of aligning IT and business capabilities. As modular approach emphasizes performance, it places utmost importance on the usefulness of knowledge. To gain the most from the captured knowledge, both structure (to define usage) and context (to promote understanding) are maintained.

The modular approach has been widely employed in mass customization. Dissemination of knowledge is IT-based and technology departments and personnel play a vital role in the

creation and maintenance of knowledge objects. Advantages of the modular approach include (1) reuse, (2) easy access to information in both time and space, and (3) consistency of knowledge across the entire organization. Disadvantages include (1) the codification process may dilute the substance of the knowledge object, (2) deciphering the context is difficult, even when it is embedded in the object and (3) this approach can lead to the creation of very large repositories leading to knowledge bloat. While offering easy and fast reuse, the modular approach may lack the effectiveness of personalized, real-time knowledge. For managing knowledge, the modular approach suggests the use of components for managing content and context (e.g., search, archival, data warehousing and mining, metadata management, linguistic/semantic analysis, workflow management), support for defining information architectures, and personalization [33].

Knowledge process is recursive, expanding, often discontinuous, and hence difficult to understand [12]. To identify the requirements for KM, it is essential to understand knowledge processes. Coordination theory has been successfully utilized to understand processes and can be applied to knowledge processes as well.

3. ELICITING PROCESS COORDINATION REQUIREMENTS

To develop coordination requirements for the KM process we must develop a process model that takes into account alternative approaches to knowledge management. We construct such a "generalized" KM process model inductively by first modeling a number of distinct approaches to KM. Ideally we would select a large number of specific KM examples from a wide range of organizational settings. Given the exploratory nature of this research we have adopted a more modest approach, selecting a smaller number of representative KM descriptions. Exploratory analysis is recommended when the boundaries between a phenomenon and a context are unclear and when the intervention being evaluated has no clear one set of outcomes [32]. It is

intuitively evident that understanding the KM process and how it is coordinated is important for knowledge sharing/reuse across KMS. However, whether additional requirements are mandated by process coordination and if so, what is the set of additional requirements is not known. Hence we adopt an exploratory analysis focusing on a smaller number of carefully chosen examples.

Selection of Examples: Two requirements must be satisfied by each of our examples (referred to as "texts"). First, the text must be a publicly available document that has been used /cited in KM literature. Second, the text must examine KM from both strategic and technical perspectives so that we understand not only "why" knowledge was managed in a specific fashion but also "how" this management method can be implemented or facilitated. We selected four texts [14, 21, 24, 27] representative of current thinking on KM best practices that met these requirements. Two represent the modular approach and two represent the shared-experiences approach for KM.

Format for Analysis and Gathering Results: We analyze the selected texts using Text-Based Process Analysis (TBPA)[2] [30]. TBPA uses the *dependency diagram* to represent the key activities in processes and the dependencies among them. As argued in [19], there are natural advantages to using dependency diagrams to represent processes when the focus is on process innovation through better coordination. While the dependency diagram is a useful tool, we also need a method to develop and analyze it. TBPA addresses this need by providing a method to systematically develop dependency diagrams from informal process descriptions, and integrate multiple views of a process into a single generalized process model. It is a good fit because, while many methods exist to represent and analyze processes, most do not identify coordination issues and thus do not support the systematic exploration of coordination alternatives [19]. TBPA

[2] For a brief introduction to TBPA and the coordination theory on which it is based, see Appendices I and II.

offers several advantages that make it eminently suitable for this research. First, it is explicitly intended for analyzing existing textual descriptions of processes. It is a good choice when trying to sweep in a range of existing perspectives from across organizations and to build on the synthesis performed by previous research. Second, TBPA proposes a visual modeling technique based on the resource flow graph, for inductively deriving dependency diagrams. This technique supports the complex task of building up process descriptions from a rich textual description that embodies a range of levels, viewpoints, and perspectives.

3.1 Results from Applying TBPA to KM processes

There are three principle phases of TBPA as applied to the current analysis: first, a text is analyzed to produce a dependency diagram. Second, the dependency diagram is annotated with coordination information. Third, the dependency diagrams are arranged into a taxonomy. For brevity, we have deferred the details of these phases to Appendix II.

Using the TBPA method we developed dependency diagrams for the four texts analyzed, from which we derived the taxonomy shown in Figure 1. There are three top-level (most generic) processes for KM. These top level processes correspond to three strategies for managing knowledge that differ as to when knowledge is created relative to its application to a new problem. An analogy might be drawn to the distinction between manufacture-to-inventory and manufacture-to-order in operations management:

CREATE REUSABLE KNOWLEDGE OBJECT (CRKO) is an approach which creates knowledge before it is needed (manufacture-to-inventory). Knowledge is packaged in "knowledge objects" and stored in a knowledge repository. A knowledge object stores a detailed description of the knowledge along with added structure such as keywords to permit searching. A key issue is to anticipate in advance what sorts of fields, keywords, and other structures will be most useful to those who will later seek this knowledge.

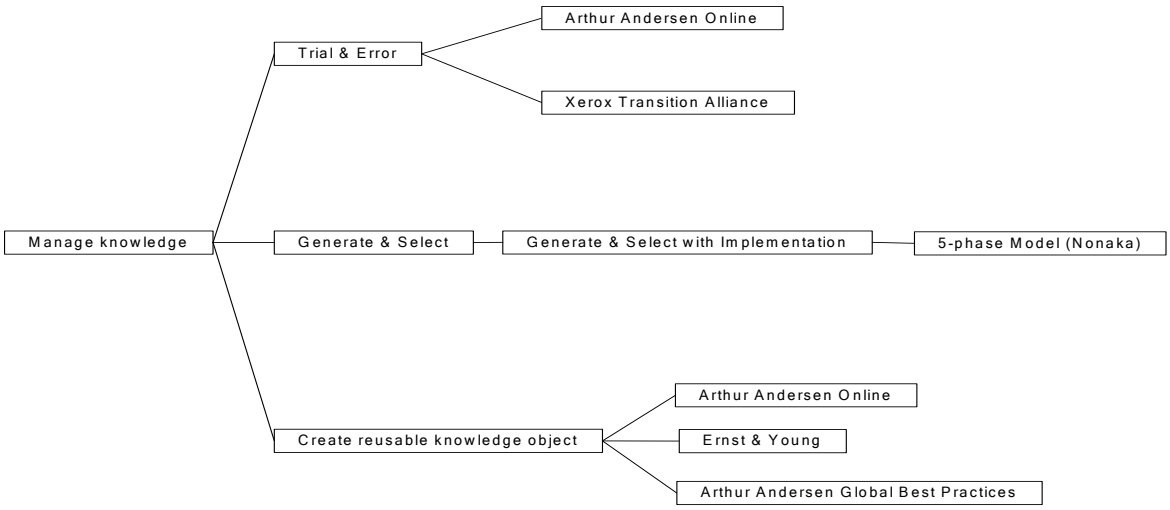

Figure 1: Taxonomy for Knowledge Management

GENERATE AND SELECT (G&S) is a process that creates knowledge when needed (manufacture-to-order). Knowledge is gathered by presenting a problem, typically to stakeholders or experts, and having participants generate a range of alternative responses. A solution is selected from among the alternatives. A key issue here is to communicate the details of the problem or situation to potential participants and to manage the flow of information among all concerned. **TRIAL AND ERROR (T&E)** is a process by which knowledge is created concurrently with its application (concurrent manufacturing). Knowledge is generated iteratively in the form of potential solutions which are tried out and modified based on experience. Key issues are to manage the cost of iteration and to learn enough from each.

Two of these three generic processes seem to correspond with the two approaches identified in section 2 (which informed our original selection of texts). CRKO corresponds to the modular approach and G&S resembles the shared experiences approach. T&E appears to represent a third, orthogonal dimension: the degree to which KM is iterative. These strategies are not mutually exclusive. While different firms may need to emphasize different strategies [14], in

principle an organization could emphasize more than one (even all three). Our goal is to develop a set of coordination requirements for KM that encompasses all three strategies.

3.1.1 Identify Common Elements in KM Processes

To construct a generalized KM process model we begin by examining the dependency diagrams corresponding to the three KM processes identified, for identifying common elements. The dependency diagrams coded for the three processes identified are in figures 2, 3, and 4. By examining these dependency diagrams certain common elements are identified and listed below.

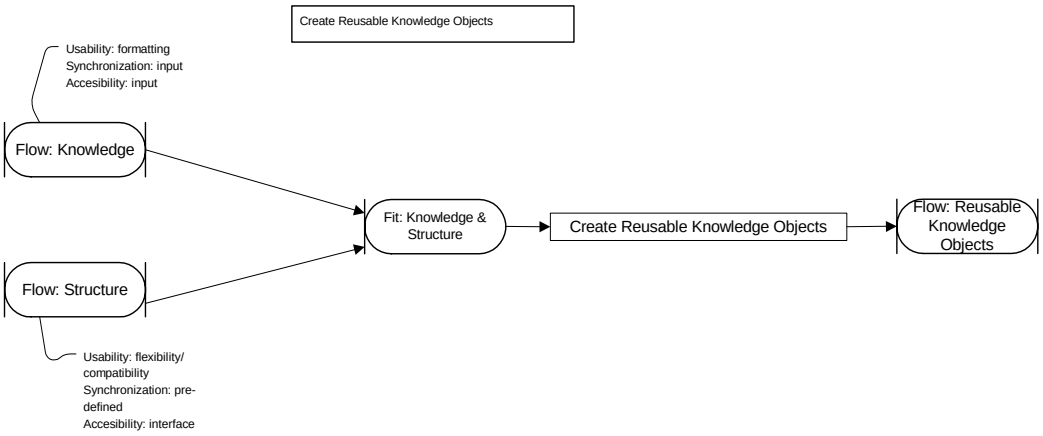

Figure 2: Dependency diagram for Create Reusable Knowledge Object (CRKO)

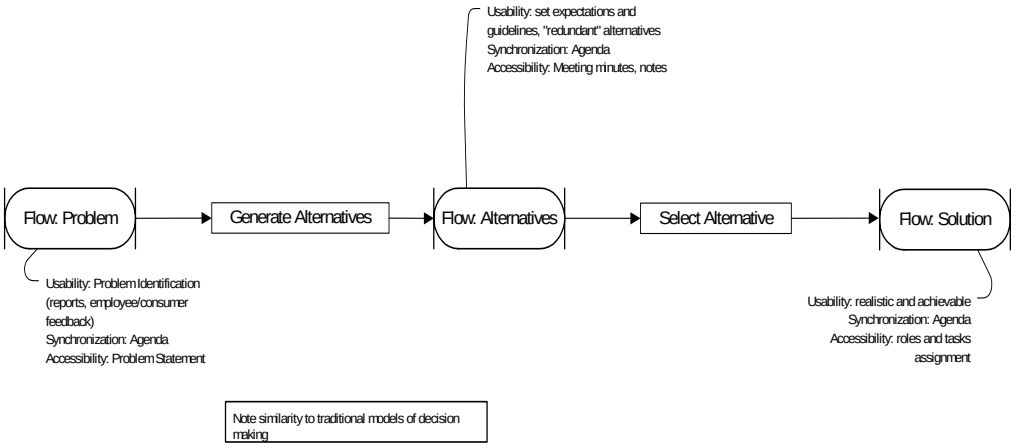

Figure 3: Dependency diagram for Generate & Select (G&S)

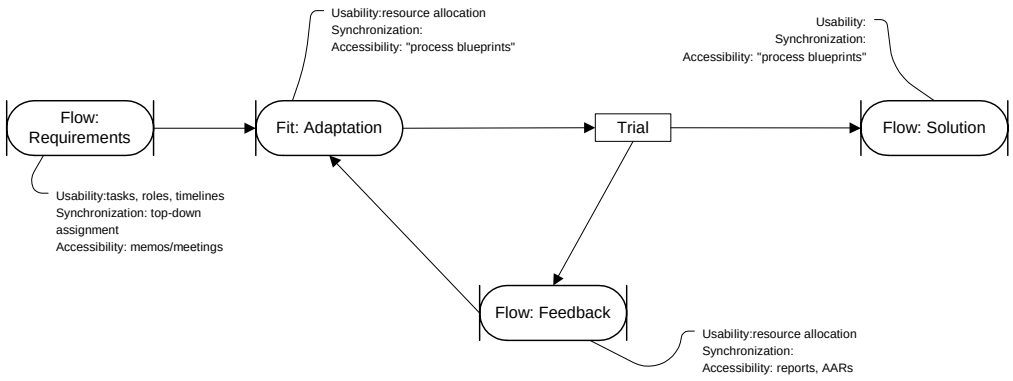

Figure 4: Dependency diagram for Trial & Error (T&E)

All the processes involve a very significant knowledge capture dependency: In the case of CRKO this is explicit. By further examining the role played by knowledge in the other two processes, it can be identified as an important element which is currently implicit in (or missing from) the dependency diagrams. G&S must capture the knowledge required to generate alternatives. Generating alternatives is really a kind of localized repository of knowledge objects. Each alternative can be viewed as a "disposable knowledge object" generated for solving this problem. The statement of an alternative is based on existing knowledge resources (in the examples which gave rise to this map this was the "personalization" approach in which the existing knowledge was held by experts who were contacted for the problem solving process). Similarly, T&E needs to capture the knowledge required to construct trial solutions. Figure 5 shows this common dependency, with G&S and T&E modified as described.

All processes respond to some purpose that defines the knowledge collection: For CRKO the purpose is embodied in a structure that captures the intended future use of the knowledge. In both other processes the framework is the requirements or problem statement. This commonality is shown in figure 6.

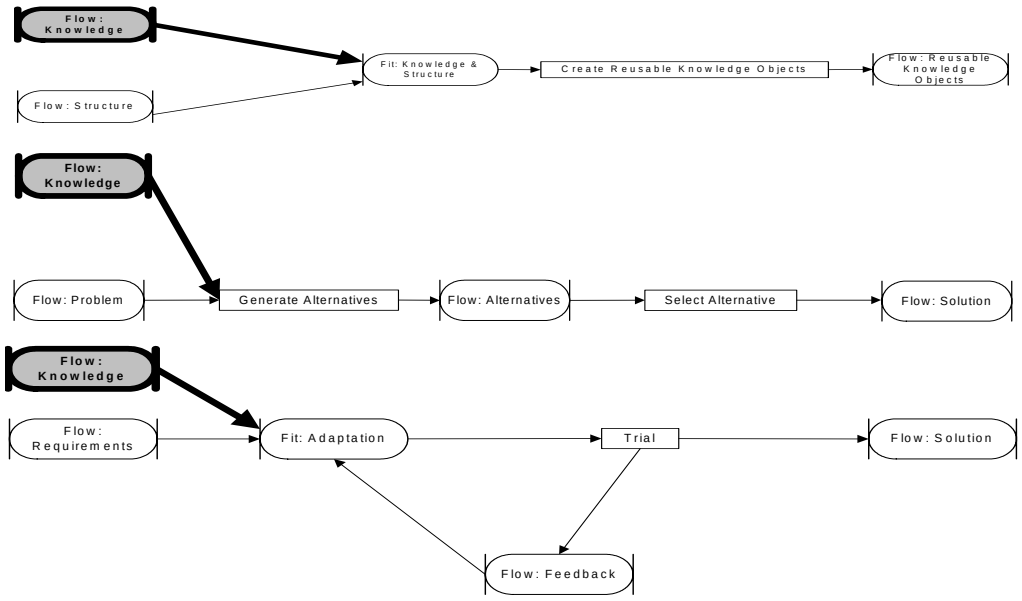

Figure 5: Common knowledge-capture dependency

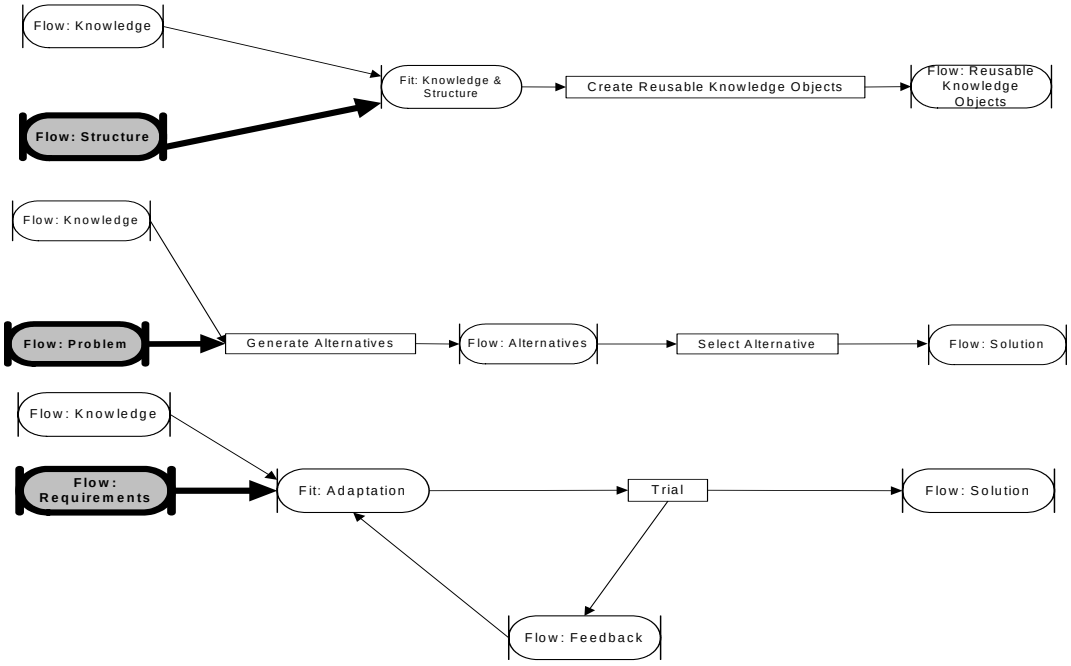

Figure 6: Common requirements definition dependency

All processes involve producing a solution in response to the given purpose: CRKO disseminates reusable knowledge objects in response to general requirements, while the other

two processes disseminate knowledge in the form of “solutions” to problems. This common element is in figure 7.

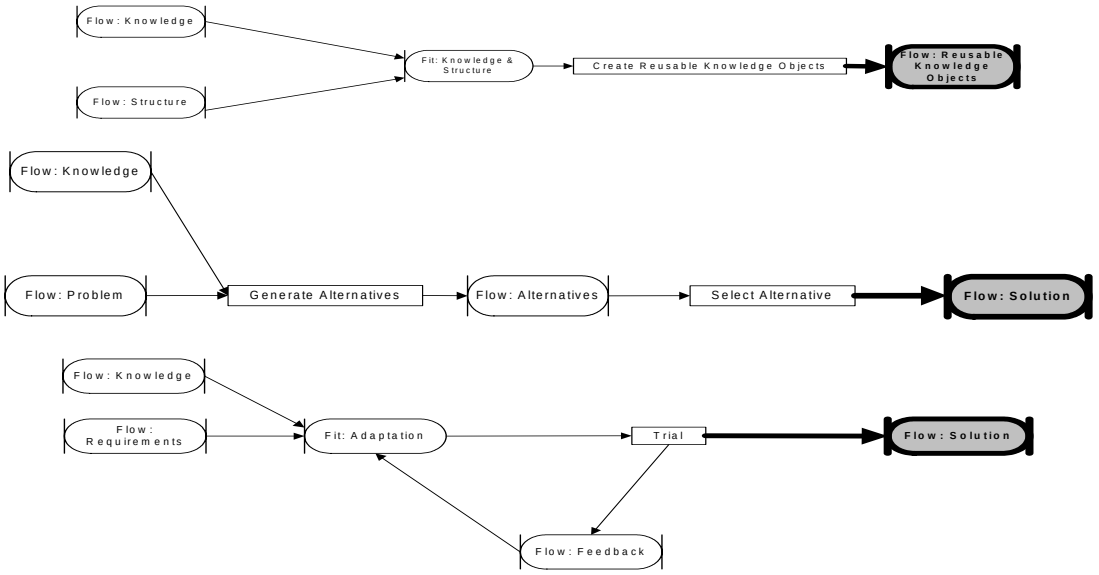

Figure 7: Common knowledge dissemination dependency

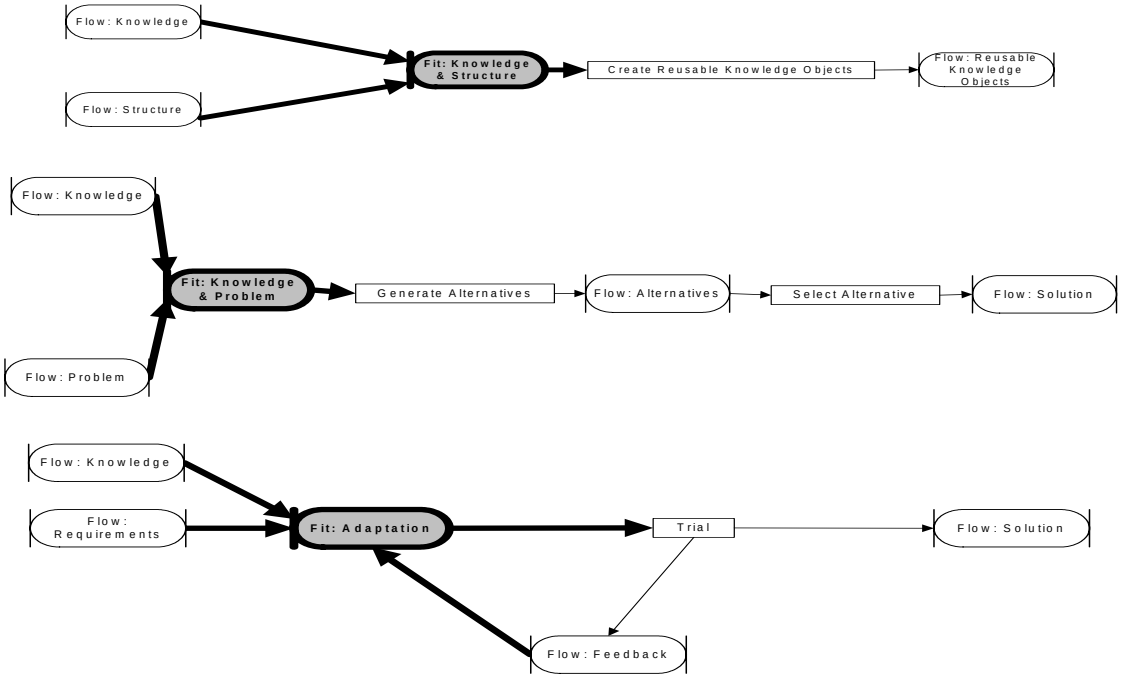

Figure 8: Common fit dependency

All processes strive to fit knowledge to purpose: In all cases a solution depends in part on fitting available knowledge to the defined purpose. Note that in the case of G&S there is no fit dependency, but rather an activity (GENERATE ALTERNATIVES). Here we argue that there is an implicit fit dependency that we render explicitly in the modified version of G&S in Figure 8.

3.1.2 A Generalized KM Process

Based on these common elements, we propose the generalized KM process shown in Figure 9:

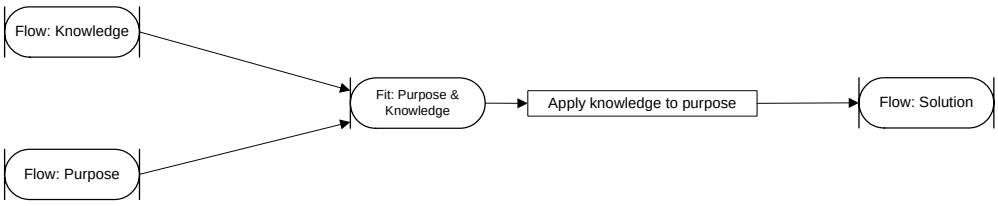

Figure 9: Dependency diagram for Manage knowledge

There are four key dependencies in this process:

- *Knowledge capture* (flow of knowledge from sources to the KM system)
- *Knowledge purposing* (flow of purpose/requirements/problem definition to the KM system, i.e. the development of such a purpose)
- *Knowledge structuring* (fitting the captured knowledge to the defined purpose)
- *Knowledge dissemination* (making the solution, and the knowledge it embodies, available to appropriate stakeholders)

4. COORDINATION REQUIREMENTS FOR DESIGNING A KMS

Having developed a generalized dependency diagram for KM, with three important variations, we now identify a set of coordination requirements for designing a KMS. These requirements exist both within a local KMS (knowledge pocket) and among the KMS distributed within the enterprise (knowledge network). Together, the resulting coordination requirements serve as the basis for the Enterprise Knowledge Architecture described in section 5.

4.1 Coordination Requirements for Managing Knowledge Pockets

To develop coordination requirements for a knowledge pocket we analyze the four key dependencies within the generalized KM process. For each dependency, we identify the coordination problem to be solved, along with a few key alternatives for addressing the problem.

Knowledge purposing (Flow from Define requirements to Fit dependency) - The knowledge purposing dependency mandates that the process clearly specify the purpose for which knowledge is being obtained. The KM process must include a mechanism for developing comprehensive and coherent requirements.

There are two distinct specializations of purpose in the three KM processes considered, each suggesting a different set of approaches for managing this dependency: for G&S and T&E, purpose is a problem to be solved, a set of requirements to be satisfied. From this perspective, the requirements definition dependency can be managed using systems analysis techniques from information systems, as well as related problem definition methods from engineering sciences. In G&S we could in principle have multiple partial views of purpose, each resulting in a different set of alternatives. In T&E we could in principle extract the purpose over multiple iterations.

For CRKO, purpose is a structure embodying a general set of requirements, independent of a specific application. Developing such a structure would seem difficult. First, we must identify the structure in advance of any specific application, and thus must anticipate a wide array of possible constraints and requirements. Second, we are not stating requirements explicitly but rather trying to embody them in a set of structural constraints on the knowledge object. As evident in the following discussion, CRKO offers powerful facilities for capturing and disseminating knowledge, but those advantages are balanced by the challenge of defining an appropriate structure for the knowledge objects.

Knowledge Capture (Flow from Create knowledge to Fit dependency) - Knowledge capture is perhaps the dependency most associated with KM. The key challenge here is to make the organization's knowledge assets available for application to the problem at hand. This dependency is especially central to the CRKO approach. In section 3.1.2 we characterized CRKO as managing part of the fit between knowledge and purpose by creating knowledge objects with a general purpose in mind. We can view the same objects as also managing the knowledge capture dependency by converting knowledge assets into a standard reusable format. Indeed, CRKO can replace the Knowledge Capture dependency in figure 9.

Knowledge Structuring -- Fit Between Purpose and Knowledge - The fit dependency requires that salient features of the problem be matched with chunks of relevant knowledge. CRKO manages the fit dependency by separating out a general set of requirements, a *structure,* which is to be applied to the knowledge. Creating a knowledge object then becomes a kind of editorial process in which knowledge is fit to this structure. The structure could be defined by the *a priori* creation of a knowledge-taxonomy or thesaurus. Using this, a "knowledge factory" can automatically sort any new information that enters the knowledge exchange into relevant categories. Another approach, identified by Balasubramanian et al. [2], is to organize information around the workflow of decision-makers. Regardless of the approach taken, organizations must be able to capture and codify content into usable chunks and perform a quality check on it.

G&S manages the fit dependency by developing a set of alternative solutions, each representing a potential fit between the problem and knowledge. Each alternative encompasses some subset of the knowledge and matches it up with some aspect of the problem. It facilitates a subsequent winnowing out of the best alternative and adapting it to solve the problem. T&E

manages the fit dependency iteratively by roughly fitting knowledge to the problem and then adjusting that fit based upon a trial of the solution.

Knowledge Dissemination - The requirement imposed by the knowledge dissemination dependency is that the specific application of knowledge developed during the KM process be made available to downstream processes that can employ that application. This dependency can be thought of as the interface between KM and knowledge deployment.

CRKO makes knowledge available for subsequent use in solving problems by publishing it as knowledge objects. Note that in publishing it, one must make it accessible to future consumers by packaging it in a readable form, store it where it can be accessed by those consumers, and index the objects so that consumers can find the appropriate object to be used. As the difficulty of searching for existing solutions is a potential roadblock to successful reuse, many of the technologies now being used for information retrieval and digital libraries are likely to be relevant for managing this dependency. In other words, knowledge selection is a critical part of knowledge dissemination: a KM system must allow selective access to the knowledge that flows thorough (and is captured in) it.

T&E distinguishes between the knowledge component of the solution and the solution itself. Each trial solution applies knowledge to the problem at hand to obtain results which constitutes newly created knowledge. This, in turn, must be fed back into the trial and error process. Thus, knowledge dissemination includes feedback, which is a dissemination of knowledge back to those who are constructing the next trial.

Finally, both T&E and G&S make use of the solution itself as a way to disseminate knowledge. Knowledge becomes packaged implicitly in the form of a practice, a solution that embodies knowledge in a practical way. This is the most useful packaging of the knowledge for

the end user as knowledge is packaged in a form that is immediately usable. However, if we are to follow the recommendations of the literature on organizational learning, it is important that this solution be linked back to the knowledge which gave rise to it, to support further adaptation of the solution and further development of knowledge concerning best practices.

4.2 Coordination Requirements for Managing a Knowledge Network

The MANAGE KNOWLEDGE process can be expanded to describe knowledge networks in several ways. In general, the input to Manage Knowledge (purpose and knowledge) can be produced by some upstream KM process(es). Similarly, the output, while termed a solution, can be considered a form of knowledge (as argued in the analysis of the T&E specialization above) and can be an input to a downstream KM process. Thus any Manage Knowledge process can be viewed as a node in knowledge network. In assembling such a network we need to manage the links between the Manage Knowledge processes, which involves managing the knowledge dissemination dependency (of the upstream process) and the knowledge capture dependency (of the downstream process). There are three alternatives to consider: (1) the coordination of the upstream knowledge dissemination dependency takes into consideration the downstream knowledge capture dependency. (2) The coordination of the downstream knowledge capture dependency takes into account the upstream knowledge dissemination dependency. (3) The two dependencies are integrated into a single knowledge transfer dependency, which allows jointly managing the upstream and downstream coordination issues. This knowledge transfer dependency brings additional requirements such as *knowledge sharing* (e.g., organizations may allow knowledge workers to build communities and social networks [29]) and *expertise identification* (e.g., organizations may support activities that help locate experts in a subject area, creation of a network of experts, build business yellow pages, and create affinity identifiers).

The other two generic KM processes also have a role to play in the creation of knowledge networks. The TRIAL AND ERROR process can enable the iterative development of knowledge assets. For example, CRKO can be subsumed in a T&E process to permit feedback from downstream users to be progressively incorporated into the reusable knowledge objects deployed in a knowledge network. GENERATE AND SELECT can take advantage of the heterogeneous knowledge resources available on a network by using such heterogeneity to generate a wider range of alternatives than would be available within the confines of a single knowledge pocket.

5. ENTERPRISE KNOWLEDGE ARCHITECTURE (EKA)

To define a context for the coordination requirements identified in this paper, we conceptualize the Enterprise Knowledge Architecture (EKA). The objective of the EKA is to promote a consistent architecture for all KMS developed, thereby permitting their integration into a knowledge network. We divide the EKA into four distinct *domains*: business process, information and knowledge, sensory, and organization [15]. These layers conceptually define the "bins" in the EKA that would hold the corresponding functional requirements for KM. For instance, the coordination requirements (addressed in this paper) are defined in the process layer. The requirements for structural integration and for physical integration (neither is addressed here) would be part of the data/information layer and sensory layer respectively.

In general terms, an EKA is an information systems architecture, where we define a set of design rules that specify the components of a system, the relationships between those components (i.e., interface design), the key design principles for each component, the master plan for IT procurement, and the plan for systems development [3, 10]. Stated differently, an EKA is an information systems architecture for a KMS.

We identify three critical requirements that the EKA must address for organizations to integrate knowledge pockets and create knowledge networks: (1) the individual KMSs (or knowledge pockets) must be *physically integrated*. (2) The *coordination requirements* for software components and business processes must be identified and managed. (3) The *structural conflicts* resulting from differences in thought-models and vocabulary must be resolved. In this paper we assume that the IT infrastructure to support physical integration is in place. The structural conflicts may be resolved by defining a metadata or a "meta-knowledge" layer. Identifying the requirements for and defining the mechanisms to manage the data in this layer is a separate research agenda on its own and is not addressed further in this paper. This leaves us with the main focus of this paper -- identifying the coordination requirements for the EKA.

5.1 EKA Illustration

To make the concepts less abstract, we describe how the coordination requirements could be used in practice. Consider a consulting company that uses a KMS to help its consultants. Had they used our methodology, they would have addressed all coordination requirements. *Coordination for managing knowledge pockets*: When a new project is started, a new KM process is initiated. This process must have a well-defined purpose - to help our client perform an effective usability analysis on their website. First, relevant knowledge assets of the organization must be marshaled for this purpose. Processes should be launched to locate, gather, communicate, surface, and store that knowledge. When submitting use cases for a project, a process should submit that use case to the repository managed by the KM system. In terms of technologies, organizations could use workflow management or content management systems to support knowledge capture. Processes should also elicit the purpose behind each scenario and capture this with the artifacts created to solve a problem.

A separate process to structure the knowledge must be launched. This structure could already exist or be created for this purpose. For example, one can identify the need for use cases, domain expertise and user feedback. This allows the KMS to store prior knowledge into these categories. A knowledge dissemination process is needed next. This will ensure that down stream processes involved in maintaining websites can access the use case findings. Depending on the approach – CRKO, T&E or G&S – these use cases will be presented to the consultants differently. For G&S, use cases will be presented in a discussion setting and consultants can refer to that and make comments or repurpose it for the new engagement.

Knowledge network requirements: Since usability testing is done on many projects, knowledge transfer must occur between old and new projects. In our case, usability experts should be identified based on the projects they are involved and, once identified, knowledge seekers should be able to contact them through their social network (using tools like LinkedIn) and discuss their problems in knowledge communities (using wikis). These communities should be supported by processes that make it easy to create these communities and share artifacts during discussions. In addition, coordination process should allow users to seek knowledge from diverse projects. For example, while the client maybe based in financial services, the current project could benefit from inputs from other industries. The availability of global knowledge and the ability to create local solutions by mixing and matching assets from other usability testing projects must exist for satisfying the network coordination requirement. As there is a strong need to iterate through the solutions, version controls tools should be part of the KM support.

The EKA thus supports a number of strategies for integrating KM processes across a knowledge network. These requirements can be summarized in Table 1 shown below.

Coordination requirements		Enabling Technologies
Knowledge pocket	**Knowledge network**	
Knowledge purposing Knowledge capture Knowledge structuring Knowledge dissemination	Knowledge transfer Integration of heterogeneous knowledge sources Intermediation of CRKO processes Iterative development of solutions based on user feedback	Work flow management Search engines Content management Transaction management Business intelligence Visualization

Table 1: EKA Process Layer Requirements

Once an organization has identified its process and data requirements, it can either custom build components or use existing technologies to build and deliver the functionality. Typically, it adopts a hybrid strategy by buying some components and building others. Regardless, it has to determine the interfaces between the Process Layer and the Information & Knowledge Layer. Once these interface requirements are known, groups within the organization can build their knowledge pockets independently and locally, while still benefitting at the enterprise level. By organizing the EKA into components and interfaces, we have created a *modular* architecture. This modularity allows the "mixing and matching" of different components to create a range of possible implementations. This will provide an organization with flexible, adaptable and evolvable platforms for better, faster and cheaper KM products.

6. DISCUSSION AND CONCLUSIONS

The primary objective of this paper is to identify process coordination requirements for designing and implementing KMS. The purpose of these requirements is to design KM systems that can be integrated with other similarly designed KMS to create a knowledge network that

enables knowledge sharing across systems. We do so by performing an exploratory analysis of four cases and applying the TBPA methodology to derive specific process coordination requirements. This paper makes several contributions. First, to define the coordination requirements we identify three fundamental needs to integrate knowledge pockets into networks: (1) providing physical integration, (2) resolving structural conflicts, and (3) defining coordination requirements. Second, we analyze existing approaches to KM and derive a set of coordination requirements. In doing so, we (1) examine KM and classify the key KM approaches; (2) analyze each approach using TBPA to create a comprehensive KM taxonomy. The first proposition is indeed supported as coordination theory was used to analyze the texts and create a taxonomy of knowledge processes from which the basic set of coordination requirements were identified. These have not been examined in literature. We believe that these are interesting and useful findings that contribute to the existing body of knowledge in KM and systems design in general. (3) used the taxonomy to derive a preliminary set of requirements for coordinating knowledge processes for knowledge networks. These requirements satsify the second proposition in this study. Third, we propose an Enterprise Knowledge Architecture that incorporates the requirements derived from the analysis. The EKA attempts to solve integration problems by defining a "standard" for building KMS. It takes into account differing approaches to KM and is based on a comprehensive set of functional requirements.

We did not use primary data for analysis which is a key drawback of this study. This can be perceived as a meta-analysis. We worked with organizations to create our own descriptions (cases) of their knowledge management processes. However, we did not to use these because it can be viewed as a biased study (analyzing process descriptions that we created). We opted to go with descriptions of knowledge management processes that were published, adopted by

organizations and more importantly, adopted by the peer community. While this does compromise the study to an extent, we do believe that the contributions of this study are important – our objective here is not to derive the process coordination requirements but *to show that these requirements are important, can be identified using coordination theory, and play an important role in the design of information systems.*

Structural integration enables communication between the different parts of the knowledge network and permits users to access and retrieve knowledge from all parts of the network in a seamless fashion. Information heterogeneity is a key issue and cannot be completely resolved unless the organization standardizes vocabulary and defines universal thought-models. Solutions for resolving data heterogeneity are appropriate for heterogeneous data integration but are insufficient for KM. For integrating knowledge repositories, it is important to not only capture the context for exchanging semantic values (or context mediation), but also to communicate to knowledge consumers how and why the context mediation process was performed. The context for a knowledge object in a knowledge repository must include meta-knowledge that answers such questions as how was it created, how was it used, how useful was it, and who is responsible, questions which are not addressed by research in data integration. These have been addressed for KM using a classification scheme for types of knowledge [16]. This constellation of strategies leads us to believe that a layered representation of the meta-knowledge in a KMS is necessary for resolving structural conflicts.

References

1. Alavi, M. and Leidner, D. Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly*, 25, (2001), 107-136.
2. Balasubramanian, P., Nochur, K., Hendersen, J.C., and Kwan, M.M. Managing process knowledge for decision support. *Decision Support Systems*, 27, 1,2, (1999), 145-162.
3. Baldwin, C. and Clark, K. *Design Rules: The Power of Modularity*. Cambridge, MA: MIT Press, 2000.
4. Bove, T. *EAI: Providing Stability in the Whirlwind of e-Commerce*. 2001 [cited March 2002]; Available from: http://eai.ebizq.net/str/bove_1.html.

5. Brown, J.S. and Druguid, P. *The Social Life of Information*. Boston, MA: Harvard Business Press, 1999.
6. Cummins, F. *Enterprise Integration: An Architecture for Enterprise Application and Systems Integration*. New York: John Wiley & Sons, Inc., 2002.
7. Davenport, T. and Prusak , L. *Information Ecology*. New York: Oxford University Press, 1997.
8. Davenport, T.H. and Prusak, L. *Working Knowledge: How Organizations Manage What they Know*. First ed. Boston: Harvard Business School Press, 1998.
9. Dellarocas, C., *A Coordination Perspective on Software Architecture: Towards a Design Handbook for Integrating Software Components*. 1996, Massachusetts Institute of Technology.
10. Dreyfus, D. and Iyer, B. *Enterprise Architecture: A Social Network Perspective*. in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS-39)*. 2006. Koloa, Kauai HI.
11. Feng, K., Chen, E., and Liou, W. Implementation of Knowledge Management Systems and Firm Performance: An Empirical Investigation. *Journal of Computer Information Systems*, 45, 2, (2004), 92-104.
12. Grover, V. and Davenport, T. General Perspectives on Knowledge Management. *Journal of Management Information Systems*, 18, 1, (2001), 5--21.
13. Hamel, G. and Prahalad, C.K. Strategic Intent. *Harvard Business Review*, May-June, (1991), 63-76.
14. Hansen, M.T., Nohria, N., and Tierney, T. What's Your Strategy for Managing Knowledge? *Harvard Business Review*, March-April, (1999), 106-116.
15. Iyer, B. and Gottlieb, R. The Four-Domain Architecture: An approach to support enterprise architecture design. *IBM Systems Journal*, 43, 3, (2004), 1-10.
16. Iyer, B., Shankaranarayanan., G., and Lenard, M. Model Management Decision Environment: A Web Service Prototype for Spreadsheet Models. *Decision Support System*, 40, 2, (2005), 283-304.
17. Linthicum, D. *Enterprise Application Integration*. Upper Saddle River, NJ: Addison-Wesley, 2000.
18. Malone, T.W. and Crowston, K. The Interdisciplinary Study of Coordination. *ACM Computing Surveys*, 26, 1, (1994), 87-119.
19. Malone, T.W., Crowston, K., and Herman, G.A. *Organizing Business Knowledge: The MIT Process Handbook*. Cambridge, MA: MIT Press, 2003.
20. Malone, T.W., Crowston, K., Lee, J., Pentland, B., Dellarocas, C., Wyner, G., Quimby, J., Osborn, C.S., Bernstein, A., Herman, G., Klein, M., and O'Donnell, E. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science*, 45, 3, (1999), 425-443.
21. March, A. and Garvin, D.A. *A Note on Knowledge Management*. Harvard Business School, Boston, 1997.
22. Nah, F., Siau, K., and Tian, Y. Knowledge Management Mechanisms of Financial Service Sites. *Communications of the ACM*, 48, 6, (2005), 117-123.
23. Nah, F., Siau, K., Tian, Y., and Ling, M. Knowledge Management Mechanisms in E-Commerce: A Study of Online Retailing and Auction Sites. *Journal of Computer Information Systems*, 42, 5, (2002), 119-128.
24. Nonaka, I. and Takeuchi, H. *The Knowledge-Creating Company*. New York: Oxford University Press, Inc., 1995.
25. Prusak, L., ed. *Knowledge In Organizations*. 1997, Butterworth-Heinemann: Newton.
26. Rhim, T.-W. and Lee, K.-H. Clustering of XML Schemas for Information Integration. *Journal of Computer Information Systems*, 46, 2, (2005), 3-13.
27. Storck, J. and Hill, P.A. Knowledge diffusion through "strategic communities". *Sloan Management Review*, 41, 2, (2000), 63-+.
28. Venkatraman, N. and Henderson, J.C. Real Strategies for Virtual Organizing. *Sloan Management Review*, (1998).
29. Wenger, E., McDermott, R., and Snyder, W.M. *Cultivating Communities of Practice*. Boston, Ma: Harvard Business School Press, 2002.

30. Wyner, G.M., *Let a Thousand Gardeners Prune: Cultivating the Distributed Design of Complex Organizations*, in *Sloan School of Management*. 2000, Massachusetts Institute of Technology: Cambridge, Massachusetts.
31. Wyner, G.M. and Lee, J. Process Specialization: Defining Specialization for State Diagrams. *Computational and Mathematical Organization Theory*, 8, 2, (2002), 133-155.
32. Yin, R. *Case Study Research: Design and Methods*: Sage Publications, 1994.
33. Zack, M. Managing Codified Knowledge. *Sloan Management Review*, 40, 4, (1995), 45-58.
34. Zlotkin, G. *Managing Shared Resource*. MIT Center for Coordination Science, 1995.

Appendix 1: An Overview of Coordination Theory

A principle benefit of coordination theory is it allows the abstraction of key decisions about how a process is organized and explores alternative ways of organizing that process. In coordination theory, a process (e.g. a business process) is represented as a set of *activities* and *dependencies* among these activities. Each dependency represents a set of coordination issues that can be managed by a number of alternate *coordination mechanisms*. We adopt the perspective, proposed in [34], that dependencies can always be framed in terms of resource flows. It follows that each dependency can be one of three types: (1) a *flow dependency* (Figure a1-1) captures the issues that arise when a resource flows from the activity that produces it to the activity that consumes it. The three key issues in managing a flow dependency are: *usability* (does the resource as produced correspond to what is needed by the consuming activity?), *synchronization* (does the resource arrive at an optimal or satisfactory time for use by the consumer?), and *accessibility* (how is the resource made available to the consumer?) (2) A *sharing dependency* (Figure a1-2) identifies the issues that arise when a single resource is consumed by multiple activities. The three key issues in managing a sharing dependency are: *resource complementarity* (are there conflicts between the requirements of the consuming activities?), *sequencing* (are there conflicts between the timing requirements of the consuming activities?), and *collision avoidance* (are there interactions between the mechanisms that provide access to the consuming activities?) (3) A *fit dependency* (Figure a1-3) identifies the issues that arise when a single resource is produced through the joint action of multiple activities. The three key issues in managing a fit dependency are almost identical to those for the sharing dependency: *resource complementarity* (are there conflicts between the outputs of the multiple producers?), *sequencing* (are there conflicts

between the timing requirements of the multiple resource flows?), and *collision avoidance* (are there interactions between the mechanisms for providing accessibility to the multiple flows?)

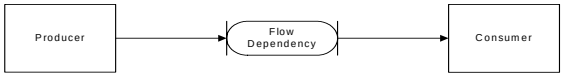

Figure a1-1: Flow dependency

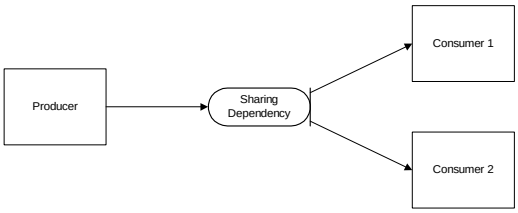

Figure a1-2. Sharing dependency

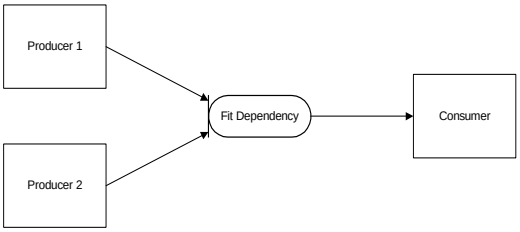

Figure a1-3. Fit dependency

Representing dependencies: The dependencies in a process are represented in a *dependency diagram* (see Figure 2 in the main paper) depicting activities and their associated dependencies.

Representing coordination mechanisms: Coordination mechanisms are represented as notes attached to dependencies in the dependency diagram. These informal descriptions will often draw upon the existing coordination literature (e.g. [9, 18, 20, 34]). While there are only three types of dependencies, there are numerous coordination mechanisms available for each. For example, Zlotkin [34] identified a set of coordination mechanisms associated with resource sharing, where the choice of mechanism depends on key properties of the resource being shared. For example, a *consumable*, *divisible* resource such as petroleum can be shared by dividing it among multiple consumers. A *non-divisible*, *non-consumable* resource such as a piece of construction equipment (a hammer) can be shared by scheduling. A *divisible*, *non-consumable*

resource such as bandwidth can be allocated by partitioning into lanes or channels. Finally (arguably, the most challenging) a *consumable, non-divisible* resource like an opera ticket can only be allocated to one of several competing parties, perhaps by lottery or managerial fiat.

Appendix II: Phases in Text-Based Process Analysis

There are three principle phases of TBPA as applied to the current analysis: first, a text is analyzed to produce a dependency diagram. Second, the dependency diagram is annotated with coordination information. Third, the dependency diagrams are arranged into a taxonomy.

Phase 1: From text to dependency diagram - The following steps are required to produce a dependency diagram:

1. Read through the text and search for point of view, activities, resources, and other components of the process. In some cases the text itself may offer a more explicit account of a process and may already have lists of activities and even resource flows available.
2. Move these components into a process fragment diagram. This map consists mainly of activities (denoted by rectangles) and resources (denoted by circles). (At this point one may also capture other components such as agents or coordination mechanisms). At this stage the activities and resources are for the most part not connected and may be placed in any arbitrary order (the placement often reflects the order in which components were encountered in the text).
3. Once the process fragment diagram is created, one begins to look for resource flows that connect different activities together. The many small fragments are replaced by a smaller number of larger fragments.
4. Search the process fragments for a point of view on the process, which may be implicit or explicit in the text. This often turns out to be a critical step in moving from the collection of

process fragments in the previous step to the *resource flow graph* described below. Clearly there may be more than one process of interest and the same process may be viewed from several different viewpoints in the process fragment diagram. One may in fact construct several process maps from a single text, or one may choose one particular point of view for a particular process and favor it over the others. What is important is that these choices and issues are all captured and recorded for future review.

5. From these efforts emerges a "resource flow graph" (e.g. Figure a1) which stands as a coherent map of the process based on the activities and resources and flows derived in the preceding steps. The derivation of a resource flow graph is not a mechanical process. While it is systematic, it draws upon the same kind of skills as are employed by someone interpreting an ancient text or seeking for meaning in a work of literature.

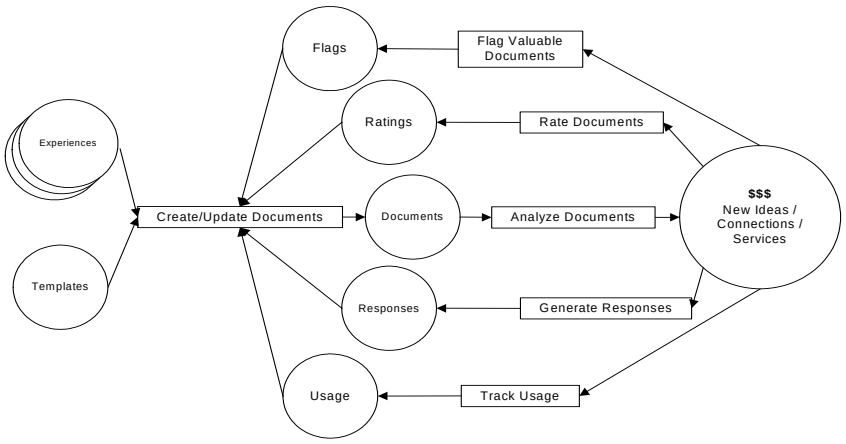

Figure a2-1: Resource Flow Graph developed from [21]

6. The resource flow graph becomes the basis for a dependency diagram. This shift involves a certain degree of abstraction in which one hides from view any resource issues that are outside the scope or beneath the threshold of the current analysis. The goal of this abstraction is to make the eventual diagram readily understandable (while maintaining its links to the context encountered during the analysis). Arriving at the dependency diagram is a

critical milestone in the analysis, since the steps that follow are essentially about annotating the diagram to represent additional issues. For example, the resource flow graph in Figure a2-1 results in the dependency diagram in Figure a2-2.

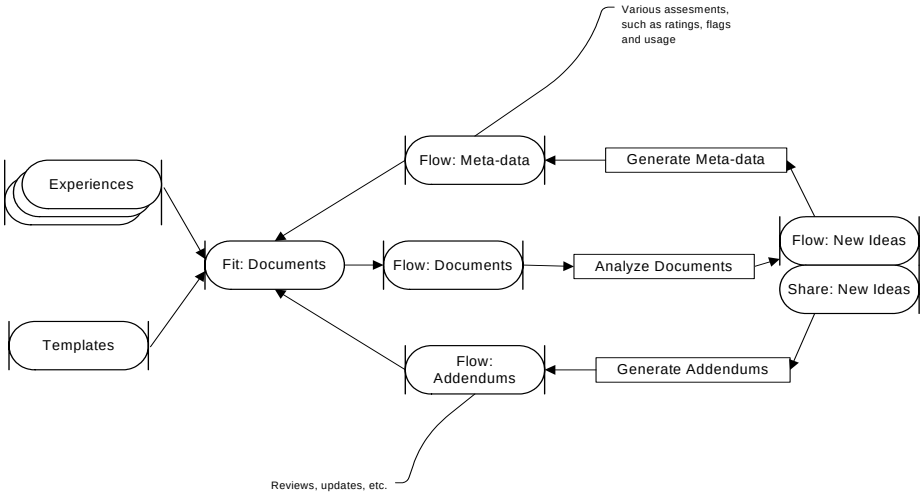

Figure a2-2: Dependency Diagram developed from [21]

Phase 2: Include coordination mechanisms - For purposes of innovation and classification, we have found that it generally suffices to annotate each dependency in the dependency diagram with an informal description of the associated coordination mechanism. The dependency diagram serves to structure and focus our examination of the text for accounts of the coordination mechanisms. For each dependency we now ask the text: how is that dependency coordinated? We are thus able to filter the text through the dependency diagram in order to pick out those portions of the story most relevant to the point of view we have chosen to bring to the text, a point of view now embodied in that dependency diagram. The reason that we retain the basic form of the dependency diagram is because our experience has shown it to be quite evocative and an exceptionally useful representation for understanding an individual process.

Phase 3: Create Taxonomy - The process taxonomy includes a collection of process descriptions and "is-kind-of" (specialization) relationships that organize these entries into a

specialization hierarchy. Process descriptions are represented in the taxonomy by rectangular boxes. Specialization relationships among process descriptions are shown as lines linking generalizations to specializations. By convention, specializations are to the right and generalizations to the left. Note that a process entry may have more than one generalization as well as more than one specialization.

Each process is added to the taxonomy by placing it under a parent process of which it is a specialization. While certain principles of specialization can be employed in this regard [31], a certain degree of judgment is also involved and thus the taxonomy inevitably reflects the point of view of its author(s). Thus as the taxonomy emerges it is important to record, as much as possible, the rationale we have employed in classifying processes. To place a process in the taxonomy, we first construct a generalization of that process and then attempt to place that generalization in the taxonomy. The original entry is then placed as a specialization of this generalization.

In placing the generalization one scans through the taxonomy to look for points of similarity. This is a relatively informal kind of pattern matching exercise. Once one has identified potential places where the entry can go, for each such place one should consider whether there is a specialization rationale that can be constructed. Note that at times the act of placing an entry into the taxonomy may suggest changes in other entries in the taxonomy or in the entry itself, or may suggest alternative process models that should also be constructed.