A Taxonomy Of Organizational Dependencies and Coordination Mechanisms[*]

Kevin Crowston
The University of Michigan
School of Business Administration
701 Tappan
Ann Arbor, MI 48109-1234 USA
+1 (313) 763-2373
Fax: +1 (313) 763-5688
crowston@umich.edu

Table of Contents

ABSTRACT

Interdependency and coordination have been perennial topics in organization studies. The two are related because coordination is seen as a response to problems caused by dependencies. Past studies, however, describe dependencies and coordination mechanisms only in general terms, without characterizing in detail differences between dependencies, the problems dependencies create or how the proposed coordination mechanisms address those problems. This vagueness makes it difficult or impossible to determine what alternative coordination mechanisms might be useful in a given circumstance or to directly translate these alternative designs into specifications of individual activities.
In this paper I develop a taxonomy of dependency types by considering possible combinations of activities using resources. The taxonomy includes task-resource dependencies and three types of task-task dependencies: shared resources, producer-consumer and common output. For each type of dependency, alternative coordination mechanisms are described. I conclude by discussing how the taxonomy helps to analyze organizational processes and suggest alternative processes.

Although you will perform with different ingredients for different dishes, the same general processes are repeated over and over again. As you enlarge your repertoire, you will find that the seemingly endless babble of recipes begins to fall rather neatly into groups of theme and variations...
--Child, Bertholle and Beck (1981, p. vii)

INTRODUCTION: DEPENDENCIES AND COORDINATION

Interdependency and coordination have been perennial topics in organization studies. The two are related because coordination is seen as a response to problems caused by dependencies. (Note that dependency and dependence mean the same thing; in this paper, I will use the first term.) For example, Thompson (1967) hypothesizes three coordination mechanisms--standardization, plan and mutual adjustment--used in response to three different patterns of dependencies--pooled, sequential or reciprocal (p. 54-55).

Most studies, however, describe dependencies and coordination mechanisms only in general terms, without characterizing in detail differences between dependencies, the problems dependencies create or how the proposed coordination mechanisms address those problems. This vagueness makes it difficult or impossible to determine what alternative coordination mechanisms might be useful in a given circumstance or to directly translate these alternative designs into specifications of individual activities or uses of information technology to support a process (e.g., as part of a business process redesign effort [(Davenport & Short, 1990; Hammer, 1990; Harrison & Pratt, 1993]).

For example, consider the process of fixing bugs in a software product, a process I recently analyzed in a search for alternative coordination mechanisms. I will use examples drawn primarily from this case site, the software development division of a minicomputer manufacturer, throughout this paper. Using Thompson's theory, we might note that programmers all contribute code to a final product, and thus have a pooled dependency and that they occasionally rely on each others' work, thus creating a sequential or sometimes reciprocal dependency. Furthermore, we might find, as predicted, that standardization, plans and mutual adjustment are all used.

This analysis leaves many questions unanswered however. For example, how else might we organize this process? What dependencies would be left (or created) if instead of dividing the work among specialists it were performed by generalists? Can we design a process which would reduce or eliminate the sequential dependencies between programmers? If not, what information do sequentially dependent programmers need to exchange and when? Would electronic mail or computer conferencing be useful to support this information exchange?

Although past conceptions of dependency do not directly address these questions, newer perspectives in artificial intelligence offer a more precise notion of dependency which might. In this paper I will draw on this work to develop a taxonomy of different kinds of organizational dependencies and associated coordination mechanisms.

Organizational Research

Dependencies and coordination have been studied by many organizational researchers. Researchers have typically conceptualized dependencies as arising between actors rather than between the tasks the actors happen to be performing. The cause of a dependency is variously viewed as control by one actor over outcomes of actions of another or due to exchanges of resources.

Litwak and Hylton (1962), for example, define interdependency as when two or more organizations must take each other into account if they are to accomplish their goals. Victor and Blackburn (1987) made this view of interdependency more precise by casting it in a game-theoretic framework. Each actor has a set of actions it could take and each actor's payoff depends on the combined choice of actions, thus the payoffs an actor gets may depend on the other actors' choice of actions. Dependency is defined by "extent to which a unit's outcomes are controlled directly by or are contingent upon the actions of another unit" (p. 490).

An alternative conception is presented by McCann and Ferry (1979). They similarly defined interdependency as "when actions taken by one referent system affect the actions or outcomes of another referent system" (p. 113), but they operationalized the degree of dependency in terms of the amount of resources exchanged, the frequency of transactions and the value of the resources to the recipient.

Dependencies usually seem to be assumed to cause problems for the actors, although Thomas (1957) notes that dependencies can be competitive or facilitative. What these problems are is rarely spelt out in detail. Instead, most authors suggest that as dependency increases, increasingly powerful coordination mechanisms are used, without specifying precisely what problems these mechanisms address. Alternately, actors might engage in actions to reduce the degree of dependency (McCann & Ferry, 1979).

Most researchers have considered only one source of dependency. One exception is Pennings (1974), who distinguished between four different sources of dependencies: task, role/position, social and knowledge. As a result, past research has focused more on describing patterns of dependencies rather than explicating the effects of a dependency on what actors can or should do.

For example, Thompson (1967) hypothesizes three patterns of dependency--pooled, sequential and reciprocal--with corresponding coordination mechanisms, which he arranged in order of increasing strength. He further suggests that organizational hierarchies will tend to cluster groups with reciprocal interdependencies most closely, then those with sequential interdependencies, and finally those with pooled interdependencies.

Van de Ven, Delbecq and Koenig (1976) identify three modes of coordinating work activities--impersonal (plans and rules), personal (vertical supervision) and group (formal and informal meetings)--and discuss situational factors, including interdependency, that might determine which are used. They build upon Thompson's (1967) view of dependency, adding a fourth, team arrangements, in which tasks are worked on jointly and simultaneous (rather than being passed back and forth). They hypothesize that, "increases in work flow interdependency from independent to sequential to reciprocal to team will be associated with... large increases in the use of group coordination mechanisms" (p. 325), which they support with data from 16 offices and the headquarters of a state agency. Mintzberg (1979) describes a similar set of coordination mechanisms: mutual adjustment, direct supervision and four kinds of standardization: of work processes, outputs, norms and skills.

While earlier work typically hypothesizes a one-to-one relationship between patterns of dependencies and coordination mechanisms, McCann and Galbraith (1981) discuss the possibility of alternative mechanisms. They suggest that coordination strategies vary along three dimensions--formality, cooperativeness and localization--and that as dependency increases, the amount of coordination necessary increases and as conflict increases, coordination strategies chosen become increasingly formal, controlling and centralized. They therefore propose a two-by-two matrix showing conditions under which organizations will choose to coordinate by rules, mutual adjustment, hierarchy or a matrix structure.

Martinez and Jarillo (1989) survey research on coordination mechanisms used by multinational corporations. They define a coordination mechanism as, "any administrative tool for achieving integration among different units within an organization" (p. 490) and list 8 mechanisms identified by past researchers: departmentalization, centralization or decentralization, formalization or standardization, planning, output and behavior control, lateral ties, informal communications and socialization. (They do not, however, define integration.) They view the selection of coordination mechanisms as a way to implement a particular "organizational pattern" (p. 502), although they suggest that companies choose a structural configuration to match their strategy and then choose the mechanisms to implement it.

There have been other attempts to develop a framework for a more detailed analysis, but these have typically focused on only one kind of coordination problem. For example, Ching, Holsapple and Whinston (1992) have the goal of developing a model of coordination to determine possibilities for information system support. They contrast networks with markets and hierarchies, but the only coordination mechanisms they consider are decomposition of tasks and allocation of subtasks to members of the network through a bidding mechanisms extended to include the reputation of the bidding company.

To summarize, most organizational conceptions of dependencies view them as arising between actors and describe patterns of actor-to-actor dependencies. Furthermore, most researchers have viewed the actors and their tasks and therefore the dependency as given and sought to identify the mechanisms used to manage dependencies, although some have suggested assigning tasks in order to create desired dependencies or minimize undesired ones (e.g., as in "administrative management theory", Simon, 1976).

Research in Artificial Intelligence and Other Fields

In contrast to most organizational researchers, researchers in the field of artificial intelligence have analyzed dependency as arising between activities. This approach has the advantage that it allows us to easily examine implications of different patterns of task assignment. Once an assignment is determined, however, we can still determine implications of a dependency for a particular actor.

Because of these advantages, I will adopt this approach in this paper.
Researchers in artificial intelligence have considered dependencies between pairs of goals or activities in detail and tried to categorize them (e.g., (Wilensky, 1983; Stuart, 1985; Decker & Lesser, 1989; von Martial, 1989). For example, von Martial (1989) developed a taxonomy of relationships, both positive, such as synergies or equality, and negative, such as conflicting use of resources or logical incompatibility. He suggested several dimensions for this taxonomy, including explicit vs. implicit and resource vs. non-resource-based interactions.
Alexander (1964, p. 106-107) suggests expressing the dependency between design variables (essentially goals of a design task) as a correlation coefficient. This view allows for both conflicting (negative) and facilitative (positive) dependencies. He notes further that some dependencies may be logically necessary or a product of physical laws, while others may simply be true in all designs in the sample considered. He therefore recommends that two variables be considered as interacting only if "the designer can find some reason (or conceptual model) which makes sense to him and tells him why they should do so" (p. 109). A fully specified design space forms a network of linked goals; this space can then be partitioned into weakly interacting subcomponents to be worked on independently.
While these researchers have catalogued dependencies, few have discussed in detail what coordination methods might be used in response to these problems. Yu (1993) does suggest specific actions that can be taken to manage different kinds of dependencies. For example, he suggests mechanisms that might be appropriate if one actor depends on another to achieve a goal, perform a task or provide a resource (p. 35).

A Framework for Studying Coordination

In this paper, I attempt to develop a richer conception of organizational dependencies by using concepts drawn from work in AI. To clarify the relationship between dependencies and coordination, I use the framework presented by Malone and Crowston (1994), who define coordination as "managing dependencies". They analyze group action in terms of actors performing interdependent activities to achieve goals. These activities may also require or create resources of various types. For example, in the case of software bug fixing mentioned above, the actors are the customers and various employees of the software company. In some cases, a group of individuals may be represented by a single actor (Abell, 1987, p. 13); for example, to simplify the analysis of a particular subunit, the other subunits with which it interacts might all be represented as collective actors. Activities include reporting a problem, diagnosing the problem, developing a fix and delivering the fix to the customer. The goal of the process in this case appears to be eliminating problems in the system, but alternative goals--such as appearing responsive to customer requests--could also be analyzed. Finally, resources include the bug reports, information about known bugs, computer time, bug fixes and so on.
Coordination problems and mechanisms. In Malone and Crowston's (1994) view, actors in organizations face *coordination problems* arising from dependencies. In many cases, these dependencies constrain how the tasks can be performed. For example, a software engineer planning to change one module in a computer system must check that the changes will not affect other modules or arrange for any necessary changes to modules that will be affected; two engineers working on the same module must each be careful not to overwrite the other's changes. Alternately, the problem might be that we want to be sure that a particular dependency exists, e.g., we want actors to choose tasks to perform that will accomplish particular goals. In other cases, the dependency provides an opportunity; for example, if the same bug has been reported by two different customers, then the company can save time by diagnosing and fixing the bug once and sharing the solution.
Coordination mechanisms may be quite specific, such as different kinds of code management systems to control changes to software, or quite general, such as hierarchies or markets to manage assignment of activities to actors (or other resource assignment problems). Malone and Crowston (1994), for example, identify several common dependencies and analyze coordination mechanisms to manage them including *goal decomposition, resource allocation* and *synchronization*,. In general, there are many different coordination mechanisms that could be used to address the same coordination problem. Therefore, a taxonomy of dependencies also serves as a way to organize coordination mechanisms. As I present the taxonomy, I will discuss possible coordination mechanisms along with the different kinds of dependency they address.

To distinguish different kinds of dependencies, I consider what the dependencies might exist between. For simplicity, I group the elements of Malone and Crowston's (1994) framework--goals, activities, actors and resources--into two categories: (1) the objects that make up the world, in particular, those resources needed to perform activities (including the actors themselves); and (2) tasks, either a goal to be achieved or an activity to be performed. In any situation there may be multiple instances of each of these elements, possibly interdependent.

Tasks. Tasks include both achieving goals and performing activities. Goals (desired states of the world) and activities (actions performed to achieve a particular state) are clearly different. However, analyzing activities and goals together makes clear the parallels between decomposing goals into subgoals to be achieved and decomposing them into primitive activities to be performed. A second advantage of this unification is that it eliminates the need to analyze all situations to the level of primitive activities performed by individuals. Treating higher-level goals as activities to be performed by a subunit allows us to analyze assignment of goals to a subunit in the same way that we consider assigning activities to individuals. Both goals and activities are descriptions of the task to be undertaken by the actor (individual or collective) to which it is assigned. For example, an outsider (or an analyst interested in other parts of a company) might think of the purchasing department as performing a "purchase materials" activity; to the members of the purchasing department, however, this is a high-level goal to be achieved by performing numerous activities, such as qualifying suppliers, soliciting bids, awarding contracts, etc.

More powerful conceptualizations of actions may help suggest the different ways actions may be interdependent. For example, a typical model of action in AI includes preconditions, (i.e., states of the world that must hold before the activity can be performed) and effects (i.e., states of the world that become true or false as a result of the activity) (Fikes & Nilsson, 1971). (Actually, this model is relatively primitive compared to those currently used in artificial intelligence research.) For example, before engineers can diagnose problems, they must know the symptoms of the problems; afterwards, they also know the diagnosis. Of course, it is possible that they will be unable to diagnose the problem or that their diagnosis will be incorrect. Additional refinements can be made to this model to handle such events without damaging this analysis, although the model will never be complete. Given such a model, it is clear that actions may be dependent on each other in several different ways, with different implications.

Resources. I include everything used or affected by activities together in this category. Things that are not somehow used or affected by an activity are not considered; if they are not involved in the activities of some actor, then they are irrelevant to the analysis of the behaviour of that actor. Resources include both material goods and the effort of actors. For example, in the case of a car company, resources include tools, raw materials, parts, partially completed assemblies, information such as designs or process instructions as well as the effort of the employees of the company. Note that actors are viewed simply as a particularly important kind of resource. I group actors and other kinds of resources together in this way despite their significant differences because many of the steps necessary in assigning tasks to actors parallel those involved in assigning other kinds of resources to tasks.

More abstractly, particular states of the world might be considered as resources: for example, if a particular tool setup is necessary for a manufacturing task to be performed, then that setup might be considered a resource, possibly to be created by some other task. Of course, there are important differences between resources. The implications of some of these distinctions for the choice of coordination mechanisms are discussed in more detail below.

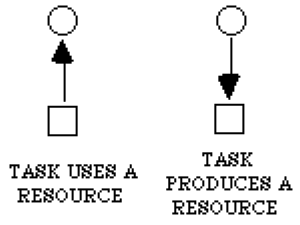
MANAGING TASK-RESOURCE DEPENDENCIES

In Malone & Crowston's (1994) framework, the important type of dependency is that between a task and a resource. Recall that tasks are either goals or activities and according to the simple model of actions, activities have preconditions and effects. (We can consider goals as having effects but no preconditions.) The preconditions and effects are the resources (or more generally, states of the world) required, consumed or created by an activity. For example, the preconditions for fixing a bug include knowing what the bug is, having access to the source code, having the capability of fixing bugs, etc.; the effect, having a patch that fixes the code. In turn, integrating a patch requires knowing the patch, having access to the object code of the entire system, etc. and results in a new system.

These dependencies might be shown graphically as in Figure 1, omitting for the moment possible difference in the types of resources.

FIGURE 1

Tasks use or produce resources.



Task Uses a Resource

Consider first a task that requires some resource. If there is just one appropriate resource known, then that resource must be the one used. This simple case includes an actor deciding that it should perform a task itself or knowing only one other actor that could perform it. For example, when customers have problems with the operating system, typically their only choice is to report the problem to the response centre and ask them to fix the bug. (Of course, customers would like more options; for example, if they knew the phone number of a developer, they might try to call that person directly; if they had access to the source code, sophisticated customers might try to fix certain bugs themselves.)

More commonly, however, there are many possibly appropriate resources, creating the problem of resource assignment, i.e., knowing a task with a precondition and looking for an appropriate resource that satisfies the precondition. In the remainder of this section I will consider the steps necessary to assign a resource when there is a set of resources, one of which might be appropriate. I mostly discuss the problem of assigning a task to a particular actor to be performed, but most of the steps discussed are necessary to assign any kind of resource to a task.

In order to assign resources to a task, the following steps must be performed:

1) identifying what resources are required by the task;
2) identifying what resources are available;
3) choosing a particular set of resources;
4) assigning the resource (in the case of an actor, getting the actor to work on the task).

(This is essentially an outline of a decision process--intelligence, design and choice--where the first step is divided into intelligence about the needs of the task and about the available resources.)

In principle, these steps can be performed in any order. For example, tasks can be chosen that can be performed with resources available (and that achieve higher level goals). For example, one manager interviewed suggested that in software development, a manager might divide a system into modules based on the abilities of the programmers who will work on them, rather than on some *a priori* division of the problem. A garbage can model might suggest that all steps go on simultaneously and occasionally connect more-or-less by chance (Cohen, et al., 1972). For convenience, however, I will discuss the steps in the order listed.

Identifying necessary resources. First, the resources needed by a task must be identified. For example, determining in which module of the system a bug appears identifies the resources needed by that task (i.e., an engineer with expertise in that module). In some cases the assigner may need to know what kind of resources are available to be able to characterize the task requirements along the same dimension. For example, actors' roles may range between two extremes: specialists and generalists. In a specialist model, only one actor can perform any given task, so the needs of the task must be identified in terms of these actors' specializations. In the generalist case, any actor can perform the task. In reality, things are rarely so precise, so that organizations will be located along the spectrum between these two extremes.

Identifying available resources. Second, a set of appropriate resources must be identified. In the simplest case, there is only one potential resource, for example, only one actor, that can perform the task. This may be due to specialization of roles of departments or individuals (i.e., who is supposed to do a particular task) rather than a distribution of ability (i.e., who is capable of doing the task). In the general case there may be several resources that could be used for the task, making it necessary to choose one. The available resources may be known *a priori* to the assigner; the assigner may know a larger set of resources, some of which may be appropriate; or the assigner may have to spend some effort identifying what resources might be appropriate (e.g., by asking someone else).

Choosing a resource. Third, one particular resource must be chosen. To choose a particular actor or other resource from those available requires some way to evaluate how good a particular resource will be for the task. Obviously, there are many possible bases for making this decision, such as speed, quality, availability, motivation, etc. For example, assigning bug reports to the next free engineer is a way to choose a particular resource based on availability rather than expertise. Which criteria to use depends on the nature of tasks being coordinated.

In some cases, it is important to consider when the resources are available, especially if multiple resources are required at the same time. In this case, the choice depends on when the required resources are free. Numerous techniques have been developed to schedule multiple resources. For example, Ripps (1991) presents a taxonomy of mechanisms for task synchronization in a real-time programming system which includes numerous mechanisms for transferring information between tasks or avoiding conflicts over resources. Sen (1993) discusses coordination mechanisms that might be applied to distributed resource scheduling, including contract-nets (Smith & Davis, 1981), distributed search and multi-agent planning.

Assigning resources. Finally, the assignment of the resource must be communicated to the actor performing the task. As well, for non-shareable resources, the resource must be marked as "in use" or other assigners warned to avoid conflicting assignments. When the resource is the effort of an actor, the actor must be told to perform the task. Where the personal goals of the individual actors differ significantly (e.g., when the interaction is non-routine or when the actors are whole firms rather than individuals), the assigner may have to convince the performer to do the task by designing appropriate incentives schemes or monitoring performance. Kraus (1993) discusses techniques for obtaining cooperation in non-cooperative distributed problem solving systems. In addition, as Whyte (1948) reminds us, "origination of action" is often complicated by the relationship or status differences between the two individuals.

In other cases, such effort will be unnecessary. For example, if employees are asked by a legitimate requester to do a task that fits their definition of their job, they are likely to simply do it. (Most task assignments in the companies I studied had this character.) Computer actors might not be programmed to refuse requests. In other cases, of course, the conditions had to be negotiated, as with bidding for a contract with an external company. Differences in these important motivational features will presumably affect an assigner's choice of performer.

Example resource allocation mechanisms. Different coordination mechanisms for resource assignment can be analyzed as particular choices for these four steps. In a hierarchy, the available resources are owned by the organization; if the resources are specialized, there may be only one appropriate for a given task. If the resources are generalized, the choice between them may be made based on factors known to the assigner, such as workload, or be delegated to the group. In a market, the available resources are those competing in the market place. Appropriate resources are identified through advertising or requesting bids and the choice between them made based on the bids submitted by the interested resources. In a network organization (Ching, et al., 1992) the resources are those that belong to the network; typically each member has a particular specialization it brings to the network. The basis for assignment is reciprocal relations, rather than contracts or hierarchy. These alternatives are summarized in Table 1.

TABLE 1

Decompositions of different mechanisms for resource allocation.

Step	*Market*	*Hierarchy*	*Network*
Identify needs	Based on specializations in market	Based on specializations in firm	Based on specializations in network
Identify resources	Broadcast a RFP and wait for replies; check advertising	Use known set of resources in firm	Use known set of resources belonging to network
Choose resource	Evaluate bids	Specialization; workload	Specialization
Assign			

resource	Contract	Employment relation	Network membership

Task Produces a Resource

Another kind of coordination mechanism is to choose tasks that have a particular effect, thus maintaining a goal-resource relationship between the desired resource and the tasks. To choose the tasks is a planning task, for which many methods have been investigated (e.g., (Allen, et al., 1990). In general, an actor must know the effect to be achieved and know (or be able to generate) possible tasks (subgoals or primitive activities) and their preconditions and effects, and be able to choose among multiple possible decompositions. As well, the actor may have to choose tasks that create needed resources (i.e., that maintains a precondition dependency, discussed below), for example by performing a means-end analysis.

For example, an engineer with a large change to implement might decompose the work into smaller changes made to several different parts, e.g., to different modules of the system, and then work on each of those changes independently. Similarly, process engineers decompose a design into specific operations the assembly workers can perform to assemble the cars. There are obviously many ways to decompose goals into activities. Alternately, an actor might proceed one step at a time, choosing an activity that appears to move closer to the desired goal, performing it and then reassessing the situation.

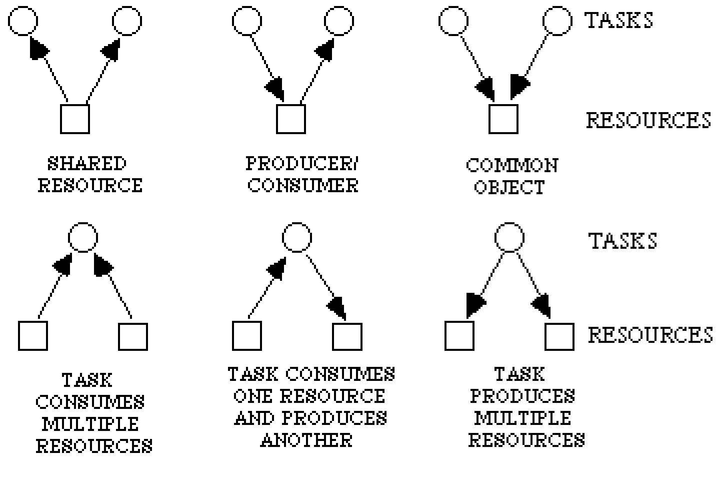
If multiple subtasks are performed to accomplish some effect, it may be necessary to integrate their results. This integration step is frequently viewed as a kind of coordination task. However, I believe that integration can be viewed as simply another part of performing the task, that is, a task is decomposed into multiple subtasks, one of which may be to integrate the results. For example an engineer who decomposes a task and requests changes from other engineers may be responsible for grouping the changes together to be submitted An integration group may be formed to integrate changes to various modules into the final system. This group knows how to recompile and link the different modules into a working operating system and how to test the resulting system.

MANAGING DEPENDENCIES AMONG MULTIPLE TASKS AND RESOURCES

In general, of course, there are multiple tasks and resources to be considered. The view proposed in this paper suggests that the only way two tasks can be dependent is via some kind of common resource. Figure 2 shows the set of possible dependencies between two tasks (the circles) and resources (the boxes); the arrows indicate flows of a resource that are either produced or used. In this framework there are three ways two tasks might have something in common and therefore three major kinds of dependencies: overlapping effects, overlapping preconditions and overlapping effects and preconditions. Similarly, a single task might use several resources, consume one resource and create another or create multiple resources. Each of these cases will be considered in turn below.

FIGURE 2

Dependencies between multiple tasks and resources


TASKS
RESOURCES
SHARED RESOURCE
PRODUCER/ CONSUMER
COMMON OBJECT
TASKS
RESOURCES
TASK CONSUMES MULTIPLE RESOURCES
TASK CONSUMES ONE RESOURCE AND PRODUCES ANOTHER
TASK PRODUCES MULTIPLE RESOURCES

Shared Resource

Two tasks are interdependent if both have the same resource as a precondition, which might require additional work to share that resource. Clearly the nature of the resource will determine what additional work is necessary. I consider in particular two dimensions along which resources differ:

shareablity and reusability, as shown in Table 2.
Shareablity describes how many tasks can use the resource simultaneously. Most resources--raw material, tools or effort--are non-shareable. Note that an actor may be assigned to multiple activities, but works on only one at any instant. Information and other states of the world are important exceptions, since multiple tasks can use the same resource if it is not changed by the tasks.
Reusability describes how many tasks can use the resource over time (von Martial, 1989, p. 62). Some resources, such as tools or information, can be used and reused; others, such as raw materials can only be used once. Of course, tools do eventually wear out; the key distinction is whether another task will be able to use the resource if it waits, as with tools, or if it the resource will be gone, as with most raw materials. (Note the no resources appear to both shareable and consumable.)

TABLE 2

Examples of resources classified by shareability and reusability.

	Shareable	*Non-shareable*
Reusable	Information: designs, problem reports, fixes	Tools: test systems, meeting rooms
Consumable		Raw materials: components, assemblies

If the common resource is shareable, then there is no conflict for two actors to use it at the same time. For example, two engineers can use the same piece of information without any problem (although there may be conflicts for the physical media on which the information is stored, a non-shareable but typically reusable resource).
If the resource is not shareable, then the two tasks can not be performed simultaneously with the available resources. Additional resources must be acquired or one of the tasks selected to be performed. If the resource is reusable then the conflict can also be resolved by performing one of the tasks at a different time. Note in particular that this applies to an actor performing multiple tasks: the actor needs to pick an order in which to do the tasks (for example, by prioritizing them and doing the most important or urgent ones first).
Managing a shared resource dependency frequently requires additional work. Since it is similar to the "mutual exclusion" problem in computer science, many of the solutions developed for that problem may be applicable. First, the conflict must be made visible, typically by marking the resource as being in use. For many physical resources, the act of using it may be sufficient (e.g., sitting in front of a computer terminal or in a conference room will prevent anyone else from using it). For less tangible resources, such as information, more elaborate schemes may be necessary. For example, a code check-out system prevents two engineers from modifying the same source code module at the same time; most databases provide some kind of locking mechanism to prevent concurrent updates to a piece of information. If the resource is not marked as being used, the performer of the task can so mark it and then use it. If it is marked as being used, the performer must either monitor the status of the resource until it becomes free or arrange to be notified when the first task is completed. Alternately, the use of the resource might be scheduled ahead of time and both tasks performed according to the schedule. For example, conference rooms and other facilities are frequently allocated with a sign-up list.
The coordination mechanisms we have discussed so far are used to avoid conflicting use of a resource. Conversely, one task might require the concurrent execution of another task , i.e., several tasks might have to be performed at the same time. Malone and Crowston (1994) call this relation a simultaneity dependency. For example, all attendees of a meeting must attend at the same time and meeting scheduling is certainly a common coordination problem. We might model this situation as several "attend meeting" activities joined with a simultaneity constraint, but it seems easier to model it as a "hold meeting" activity which requires multiple resources simultaneously. (Coordination methods for assigning multiple resources are considered below.) Similarly, two people lifting a piano must lift their ends simultaneously, but this activity is probably best modelled as a lifting action which requires multiple people. This approach eliminates the category of simultaneity constraints but for some cases might require the conceptual creation of an unnatural aggregate task.
Alternately, it may be necessary to ensure that two tasks both use the same version of a resource. For example, two designers working on the detail design of components of a product should be using the same version of the overall design. (Note that consumable resources can not be used by different

tasks at all and so are not subject to this problem.) If we consider different versions of a resource to be different resources, then the problem is to choose resources to be used by the tasks to ensure that there is a shared-resource dependency between them. This problem seems very hard. Solutions include destroying all obsolete resources, to ensure there is only one resource, checking the status of the resource or making a new copy of the resource prior to each use, or tolerating a certain level of disagreement and repairing problems after the fact.

Producer-Consumer

If a resource is the effect of one task and a precondition of another then there is a precedence dependency between the two tasks, requiring that the tasks be performed in the correct order and the flow of the resource between the two be managed. This relationship frequently holds between steps in a process. For example, in software changes, fixing the bug has the effect of creating a patch which is a precondition for integrating the complete system.

Crowston and Malone (1994) point out that precedence dependencies imply additional constraints on how tasks are performed, such as a usability constraint--ensuring that what the first task creates is in fact what the second needs--or an inventory constraint--ensuring that the resources needed by the second task are available when needed. Usability constraints can be managed by standardization, by negotiation between the user and creator or by giving the creator additional information about the needs of the user. For example, in an engineering context, design and manufacturing engineers might work together to develop products that are easy to manufacture or designers might be trained in the requirements of manufacturing processes ("design for manufacturability"). Quality control tasks can be seen as a way of ensuring that an output of one task is in fact the correct input for the next. Many other kinds of approval processes may also be serving this function. Inventory constraints can be managed by adding buffers between the two processes to smooth the flow of material or tying the rate of production to the rate of use, as with the "producer/consumer" problem in computer science or just-in-time manufacturing.

Alternately, a resource required by one task may be inadvertently or necessarily destroyed by another, what is called "clobbering" in the planning literature (Sussman, 1974, p. 116). In construction, for example, installing one piece of equipment may block access needed to install another. Fixing the clobber can be done by reordering the steps to avoid this problem or possibly adding an additional task to recreate the resource or state of the world.

Common Object

The effects of two tasks may be the same resource. This dependency can have either positive or negative effects, which requires additional effort to exploit or avoid. If both tasks do the same thing, i.e., create the same resource, then it may be desirable to merge the two tasks, reusing the resource or taking advantage of economies of scale in its production. To exploit these possible synergies requires additional activities, such as checking for duplication before one or both of the tasks have been performed and distributing the output. For example, the same problem may be reported to a software company multiple times. Rather than fixing and refixing the same problem, the customer service centre and marketing engineers check if a reported problem is in a database of known problems. If it is, then the already known solution to the problem can simply be reused, thus eliminating a task which would create a duplicate resource (the bug fix).

If two tasks specify different aspects of a common resource, then it may be necessary to add an additional task of negotiating a mutually acceptable output. For example, the interface between two modules, a common object created by the design of the modules, may be negotiated by the engineers developing the two modules. Finally, if the effects conflict, for example, both doing a task and not doing it, then it may be impossible to perform both tasks. Possible resolutions of this dependency are interestingly similar to the case where two tasks both require the same non-shareable resource, discussed in more detail below: either abandoning one task or scheduling them so they do not have to be achieved at the same time.

Dependencies Between One Task and Multiple Resources

The three cases where a task is dependent on multiple resources--either using two resources, producing two resources or using one resource and producing another--seem to be less problematic than the cases discussed above. Of the three, only the first seems to pose difficulties. In this case, there is a need to synchronize the availability of multiple resources. For example, multiple people attending a meeting can be modelled as a "hold meeting" task that requires several peoples' efforts simultaneously, as discussed above; most production activities require an actor, raw materials and tools simultaneously.

This dependency can be managed by permanently assigning all resources, or all resources but one, to the task. For example, a production activity might always be performed on a particular machine, which is always operated by a particular actor. In this case, this dependency reduces to a task using a single resource, as discussed above. More generally, some of the resources may be used by multiple activities. In this case, the dependency is typically managed by scheduling the use of the resources, as discussed above. As well, computer scientists have developed algorithms for assigning resources to avoid deadlock (a condition where one task is waiting for a resource held by the another, which in turn is waiting for the first) and starvation (where a task waits forever for the resources it needs to become available).

Multiple Modes of Use

A resource may appear as both a precondition and effect of some task; for example, modification or consumption of a resource can be modelled this way. The resulting dependencies are the combination of the dependencies from the individual operations. Two engineers who both want to modify a single software module (a shareable/reusable resource) must manage both precondition-precondition and effect-effect dependencies. From the previous discussion we can see that the precondition-precondition dependency causes no conflict (they both can read the module freely), but the effect-effect dependency might. If they are both making the identical change, then one of the tasks can be eliminated; otherwise, they will have to negotiate to ensure the resulting module is acceptable to both (i.e., that both bugs are fixed). Alternately, the module could be viewed as a non-shareable/reusable resource. In this case, the precondition-precondition dependency must be managed, e.g., by scheduling the engineers' use of the module so one makes changes and then the other.

DEPENDENCIES BETWEEN TASKS OR BETWEEN RESOURCES

In developing this taxonomy, I considered only dependencies between tasks and resources. An alternative model would include dependencies that arise between tasks or between resources. In particular, both tasks and resources can be thought of as forming decomposition hierarchies: higher-level tasks can be decomposed into subtasks, and an object into components.

Given such a model of a task, planning might be viewed as a way to manage the relationship between tasks and subtasks, that is, a way to choose a set of activities that accomplish a desired task. However, there does not appear to be much difference between this view and the view proposed in this paper in the set of coordination mechanisms applicable. It is probably conceptually simpler to minimize the number of dependencies considered.

It is also possible for different resources to be interdependent, for example, by being connected together in some kind of assembly, such as the parts of a car or of a computer system. These interdependencies are clearly important: an essential part of change management, for example, is managing the interfaces between parts to ensure that changes to one part do not interfere with the function of another.

It is probably simplest to think of these interdependencies as forming components into a larger resource and then analyzing the dependencies as created by that resource. In other words, if two tasks use different resources that are interdependent in this way, then the two tasks can be analyzed as both depending on a larger common resource. Conversely, two tasks may appear to be using a common resource because they are each dependent on components of a more complex resource. For example, two cooking tasks may both appear to consume an orange, because one requires its rind and the other its pulp. Schelling (1978) describes numerous instances where an activity depends on the macro properties of a system (e.g., the racial composition of a neighbourhood) and affects it indirectly by changing a subcomponent of the system (e.g., one person deciding to move in or out of the neighbourhood).

Nevertheless, it is clear that to manage these dependencies, actors must first identify that they exist. For example, for engineering change management, engineers must spend some effort to identify which other engineers need to be informed of a proposed change to a part. Alternately, an actor may have to choose or create another resource that maintains the given dependency. For example, in designing a large system, engineers may need to choose component parts that maintain desired relations.

DISCUSSION

The framework presented here makes a theoretical claim about the design of organizations: given a coordination problem caused by a dependency, some coordination mechanism is necessary to manage it. This claim has implication for the analysis and design of organizations. To analyze an organizational process, it is important to identify the dependencies that arise and the coordination mechanisms that are used to manage those dependencies. Fortunately, as Simon (1981) points out, in practice "most things are only weakly connected with most other things; for a tolerable description of reality only a tiny fraction of all possible interactions needs to be taken into account" (p. 221), what he calls the "empty world hypothesis".

To design a new process, it will be useful to consider alternative coordination mechanisms that could be used to manage those dependencies. One question I posed at the beginning of this paper was, in what ways can a given organization be arranged differently while achieving the same goals? Understanding the coordination problems addressed by an organization suggests alternative coordination mechanisms that could be used, thus creating a space of possible forms. It may be useful to look for more systematic approaches to searching this design space. Several strategies are apparent. As a concrete example, I will discuss the way problem reports are assigned to software engineers to be fixed and consider other ways the task could have been organized.

One approach is to identify the basic steps of the task to be performed and add coordination methods to manage problematic dependencies. For example, we might decide that primary goal of the change process is to develop a patch to fix a set of symptoms, which involves at a minimum diagnosing the bug, writing new code and integrating that code with the rest of the system. However, the person who initially knows about this task, the user of the system who has encountered a problem, is usually unable to perform this task because of a lack of skills, correct tools, etc. Therefore, there is a problematic dependency between the task and those resources, requiring a resource assignment coordination mechanism to find the appropriate resources. For the customer, a simple mechanism can be chosen, since the only other actor the customer knows about is the company's support group. This analysis can be then repeated for the other activities and actors involved.

As well, alternative assignments of abilities can be considered. For example, customers could be given the means to do some of the work themselves; for example, some companies provide customers access to their database of known bugs, allowing the customers to see if their problem has already been reported and perhaps find a solution. Facilitative coordination mechanisms can also be added. For example, since one possibility is that this task is actually a duplicate of some other task, it might be worth checking to see if a solution is already known. In the company I studied, this check was performed initially by the service centre staff, but in principle, anyone could do it (including, as discussed above, the customer).

A second approach would be to start with existing organization and modify it incrementally by substituting different coordination mechanisms for those currently used. As well, one could look for redundant steps or mismatches between tasks and actors' skills. For example, many of the actors in the mini-computer manufacturer perform some part of a task assignment mechanism to assign the task to an engineer with the appropriate skills. In other companies (and other parts of the same company) engineers are generalists rather than specialists and task assignment is based on workload rather than expertise. This system has the advantage of spreading the workload more evenly and reducing the need for engineers to coordinate changes for one bug, but does not promote or take advantage of any expertise engineers might have.

A more radical change might be to use some kind of market-like task assignment system for problem reports. In this case, a description of each problem report would be sent to all available or interested engineers. Engineers who could fix the bug would submit a bid, saying how long it would take to fix the bug, how much it would cost or even what they would charge to do it. The lowest bidder would be assigned the task. This system would have the benefits of both specialist and generalist systems: at any point, the best person available would be given the task.

The analysis of the coordination mechanisms also shows what is necessary to perform this kind of task assignment. First, assigners need to know which actors are potential performers and be able to communicate with them. Second, there must be a common language in which the tasks can be

described. Finally, the assigner needs some way to evaluate the bids returned. For their part, the performers need to be able to evaluate each task and estimate how long it would take to fix the problem (or how much it would cost) and to communicate with the task assigner.
Additional substitutions might be made possible by the use of information technology. For example, while market-like mechanisms have theoretical performance advantages, the disadvantage is that there are expensive to run: each report must be read by multiple engineers, bids must be collected and evaluated, etc. However, many of these steps could be automated, thus reducing the cost of the scheme, perhaps enough to make it attractive. For example, engineers could use rules to filter out desirable or undesirable tasks (Malone, et al., 1989); bids could be automatically collected and tasks assigned.
Many coordination mechanisms create the need to access or update information, such as a sign-up list for a resource or a centralized list of reported problems and bug fixes. Traditionally such information is kept on paper and can therefore be kept up-to-date only be in one place, either centralized, which makes choosing among multiple resources easier or decentralized, which makes accessing a particular resource easier. Either approach requires additional work to share the resources. Information technology may make it possible to do both.

CONCLUSION

In this paper, I present a taxonomy of dependencies and associated coordination mechanisms. This taxonomy is based on a simple ontology which includes resources (actors or other objects) and tasks (activities or goals to be accomplished). For simple task-resource dependencies, I present a framework for analyzing the steps in resource assignment mechanisms. More complex dependencies are analyzed by considering how a common resource is used by the two tasks or how one task can use multiple resources. The resulting dependencies and coordination mechanisms are summarized in Table 3.

TABLE 3

Summary of dependencies and coordination mechanisms.

Dependency	*Sample coordination mechanism*
To manage a dependency	
Task-resource	
task uses a resource	resource assignment (identifying necessary and available resources, choosing resources, marking resources in use)
Common effects	
same	look for duplicate tasks merge tasks or pick one to do
overlapping	negotiate a mutually agreeable result
conflicting	pick one task to do
Common preconditions	
same	
shareable resource	no conflict
reusable resource	make conflict visible schedule use of the resource
non-reusable resource	acquire more resources or pick one task to do
Effect of one is precondition of other	
same	order tasks and manage flow of resource
conflicting	reorder tasks or add another task to repair the precondition
To maintain a dependency	

Task-resource	
task creates a resource	pick tasks that accomplish desired goals
Common effects	decompose a task into subtasks (possibly including integration)
Common preconditions	ensure same version of resource
Effect of one is precondition of other	means-ends analysis

Evaluation of Contribution

Since a taxonomy *per se* is not a theory (Bacharach, 1989, p. 497), my primary focus has been on the theoretical constructs, namely dependencies, the problems created by dependencies and the coordination mechanisms actors use to manage those problems. Although Doty and Glick (Doty & Glick, 1994) suggest that a typology can be a theory, their analysis is of typologies of whole organizations that relate characteristics of those organizations to some dependent variable, such as organizational performance and is not directly applicable to the taxonomy presented here.

The primary evaluation of this work should be on the quality of these constructs: their validity (Bacharach, 1989, p. 497), comprehensiveness and parsimony (Whetten, 1989). It seems clear that these constructs are distinct from each other. The taxonomy probably errs on the side of parsimony since it characterizes coordination methods on the basis of a small number of factors, while obviously there are many reasons to choose what to do. On the other hand, as (Bacharach, 1989,p. 500) says, "some of the most abstract and broad perspectives on organizations, while not necessarily rich in detail, have provided a critical basis for cumulative research". As well, the taxonomy attempts to be comprehensive, in the sense that all dependencies should fit into one or another of the three categories. Clearly, additional dependencies and coordination mechanisms need to be added and the categories further refined. Finally, the taxonomy seems to be useful, as discussed above.

Future Research

Much remains to be done. The focus on processes suggests that it is important to collect many examples of processes to compare the coordination problems that arise (Malone, et al., 1993) and identify the coordination mechanisms used. Other kinds of organizations may have somewhat different kinds of problems, although there is likely to be substantial overlap. For example, do Japanese companies use a different set of mechanisms to manage engineering changes?

However, it is also important to identify the limitations of this work. The overall framework is focused on tasks. Organizations that do not perform tasks may not find these mechanisms useful. Some tasks (for example software requirements analysis for complex computer systems) seems to be more about developing a shared understanding of the tasks and dependencies as opposed to actually performing them (Kammerer & Crowston, 1993).

Even with these limitations, the initial results of this work should be useful in several ways. A better understanding of what is necessary for coordination may provide a more principled approach for designing new computer applications, for analyzing the way organizations are currently coordinated and for explaining perceived problems with existing approaches to coordination.

If better formalized, the kind of analysis sketched above could be incorporated into an expert support system. Such a system, given a task to be performed and the set of actors available, would suggest alternative ways to organize the actors or work out the implications of particular organizational designs. Based on the recipes for different coordination mechanisms, the system would suggest what coordination work each agent would do and even what kinds of support systems would be useful. By systematically exploring the space of possible coordination strategies, we may even be able to discover entirely new organizational forms, forms that are more efficient, flexible or satisfying to their members.

REFERENCES

Abell, P. 1987. **The Syntax of Social Life : The Theory and Method of Comparative Narratives**. New York: Clarendon Press.

Alexander, C. 1964. **Notes on the Synthesis of Form**. Cambridge, MA: Harvard University Press.

Allen, J., Hendler, J. & Tate, A. (Eds.). 1990. **Readings in Planning**. San Mateo, CA: Morgan Kaufmann.
Bacharach, S. B. 1989. Organizational theories: Some criteria for evaluation. **Academy of Management Review**, 14(4): 496-515.
Child, J., Bertholle, L. & Beck, S. 1981. **Mastering the Art of French Cooking**. New York: Alfred A. Knopf.
Ching, C., Holsapple, C. W. & Whinston, A. B. 1992. **Modeling Network Organizations: A Basis for Exploring Computer Support Coordination Possibilities**. Unpublished manuscript, Decision & Information Systems, College of Business, Arizona State Univeristy.
Cohen, M. D., March, J. G. & Olsen, J. P. 1972. A garbage can model of organizational choice. **Administrative Science Quarterly**, 17: 1-25.
Davenport, T. H. & Short, J. E. 1990. The new industrial engineering: Information technology and business process redesign. **Sloan Management Review**, 31(4): 11-27.
Decker, K. S. & Lesser, V. R. 1989. Some initial thoughts on a generic architecture of CDPS network control. In M. Benda (Ed.), **Proceedings of the Ninth Workshop on Distributed Artificial Intelligence**: 73-94. Rosario Resort, Eastsound, WA.
Doty, D. H. & Glick, W. H. 1994. Typologies as a unique form of theory building: Toward improved understanding and modeling. **Academy of Management Review**, 19(2): 230-251.
Fikes, R. E. & Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. **Artificial Intelligence**, 2: 198-208.
Hammer, M. 1990. Reengineering work: Don't automate, obliterate. **Harvard Business Review** (July-August): 104-112.
Harrison, D. B. & Pratt, M. D. 1993. A methodology for reengineering business. **Planning Review**, 21(2): 6-11.
Kammerer, E. & Crowston, K. 1993. **Coordination and Collective Mind in Software Requirements Development**. Unpublished manuscript, University of Michigan, School of Business.
Kraus, S. 1993. Agents contracting tasks in non-collaborative environments. In **Proceedings of the Eleventh National Conference on Artificial Intelligence**: 243-248. Washington, DC: AAAI Press/MIT Press.
Litwak, E. & Hylton, L. F. 1962. Interorganizational analysis: A hypothesis on coordinating agencies. **Administrative Sciences Quarterly**, 6(4): 395-420.
Malone, T. W. & Crowston, K. 1994. Toward an interdisciplinary theory of coordination. **Computing Surveys**, 26(1): 87-119.
Malone, T. W., Grant, K. R., Lai, K.-Y., Rao, R. & Rosenblitt, D. 1989. The Information Lens: An intelligent system for information sharing and coordination. In M. H. Olson (Eds.), **Technological Support for Work Group Collaboration**. Hillsdale, NJ: Lawrence Erlbaum & Associates.
Martinez, J. I. & Jarillo, J. C. 1989. The evolution of research on coordination mechanisms in multinational research. **Journal of International Business Studies**: 489-514.
McCann, J. E. & Ferry, D. L. 1979. An approach for assessing and managing inter-unit interdependence. **Academy of Management Review**, 4(1): 113-119.
McCann, J. E. & Galbraith, J. R. 1981. Interdepartmental relations. In P. C. Nystrom & W. H. Starbuck (Eds.), **The Handbook of Organizational Design** (Vol. 2): 60-84. New York: Oxford University Press.
Mintzberg, H. 1979. **The Structuring of Organizations**. Englewood Cliffs, NJ: Prentice-Hall.
Pennings, J. 1974. **Differentiation, Interdependence and Performance in Formal Organizations**. Paper presented at American Sociological Association Annual Conference, Montreal, Carnegie-Mellon University.
Ripps, D. L. 1991. Task coordination: Specific methods, general principles. **EDN**(March 1): 97-110.
Schelling, T. C. 1978. **Micromotives and Macrobehavior**. New York: Norton.
Sen, S. 1993. Predicting Tradeoffs in Contract-Based Distributed Scheduling, Unpublished Doctoral Dissertation, University of Michigan, Department of Computer Science and Engineering.
Simon, H. A. 1976. **Administrative Behavior** (3rd ed.). New York: Free Press.
Simon, H. A. 1981. **Sciences of the Artificial** (2nd ed.). Cambridge: MIT Press.
Smith, R. G. & Davis, R. 1981. Frameworks for cooperation in distributed problem solving. **IEEE Transactions on Systems, Man and Cybernetics**, 11(1): 61-70.

Stuart, C. 1985. An implementation of a multi-agent plan synchronizer. In **Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)**: 1031-1033. .
Sussman, G. J. 1974. The Virtuous Nature of Bugs. In J. Allen, J. Hendler & A. Tate (Eds.), **Readings in Planning**: 111-117. San Mateo, CA: Morgan Kaufmann.
Thomas, J. 1957. Effects of facilitative role interdependence on group functioning. **Human Relations**, 19: 347-366.
Thompson, J. D. 1967. **Organizations in Action: Social Science Bases of Administrative Theory**. New York: McGraw-Hill.
Van de Ven, A. H., Delbecq, A. L. & Koenig, R., Jr. 1976. Determinants of coordination modes within organizations. **American Sociological Review**, 41(April): 322-338.
Victor, B. & Blackburn, R. S. 1987. Interdependence: An alternative conceptualization. **Academy of Management Review**, 12(3): 486-498.
von Martial, F. 1989. Multiagent plan relationships. In M. Benda (Ed.), **Proceedings of the Ninth Workshop on Distributed Artificial Intelligence**: 59-72. Rosario Resort, Eastsound, WA.
Whetten, D. A. 1989. What constitutes a theoretical contribution? **Academy of Management Review**, 14(4): 490-495.
Whyte, W. F. 1948. **Human Relations in the Restaurant Industry**. New York: McGraw-Hill.
Wilensky, R. 1983. **Planning and Understanding: A Computational Approach to Human Reasoning**. Reading, MA: Addison-Wesley.
Yu, E. 1993. Modeling organizations for information systems requirements engineering. In **Proc. of Requirements Engineering `93**: 34-41. . Abell, P. (1987). *The Syntax of Social Life : The Theory and Method of Comparative Narratives*. New York: Clarendon Press.

Footnote

[*] This research was supported by the Center for Coordination Science at the Massachusetts Institute of Technology and by a fellowship from the Ameritech Foundation, through the Information and Organizations Program of the University of Michigan Institute for Public Policy Studies. The author thanks Brian Pentland, Michael Prietula and Thomas W. Malone for their comments on earlier drafts of this paper.