# Tools for inventing organizations:

# Toward a handbook of organizational processes

Thomas W. Malone, Kevin Crowston, Jintae Lee, Brian Pentland,
Chrysanthos Dellarocas, George Wyner, John Quimby, Charles S. Osborn,
Abraham Bernstein,
George Herman, Mark Klein, and Elisa O'Donnell

*Revised 10/98*

*Center for Coordination Science*

*Massachusetts Institute of Technology*

Published in *Management Science* 45(3) pp 425-443, March, 1999.

# Tools for inventing organizations: Toward a handbook of organizational processes


## ABSTRACT

This paper describes a novel theoretical and empirical approach to tasks such as business process redesign and knowledge management. The project involves collecting examples of how different organizations perform similar processes, and organizing these examples in an on-line ìprocess handbook". The handbook is intended to help people: (1) redesign existing organizational processes, (2) invent new organizational processes (especially ones that take advantage of information technology), and (3) share ideas about organizational practices.

A key element of the work is an approach to analyzing processes at various levels of abstraction, thus capturing both the details of specific processes as well as the "deep structure" of their similarities. This approach uses ideas from computer science about inheritance and from coordination theory about managing dependencies. A primary advantage of the approach is that it allows people to explicitly represent the similarities (and differences) among related processes and to easily find or generate sensible alternatives for how a given process could be performed. In addition to describing this new approach, the work reported here demonstrates the basic technical feasibility of these ideas and gives one example of their use in a field study.


## INTRODUCTION

In recent years, we have seen striking examples of process innovations that have transformed the way organizations work. Although initially uncommon and perceived as radical, ideas like just-in-time inventory control and concurrent engineering have become accepted as "best practice" (Carter & Baker, 1991). These innovative practices have clearly been beneficial, but most organizations remain in need of improvement, as suggested by the on-going popularity of "total quality management," "business process redesign," and "the learning organization." These slogans summarize ideas with real value, but they provide too little guidance about what the improved organization might look like in particular situations. They hold out the promise of innovation, but lack the details needed to accomplish it.

The gap between the need to innovate and the tools for doing so leaves us with a problem: How can we move beyond the practices of today to invent the best practices of tomorrow? And where will we keep getting new ideas for organizational processes to adapt to a continually changing world? For instance, how can we understand and exploit the new organizational possibilities enabled by the continuing, dramatic improvements in information technology? Given

time, managers and employees of companies will certainly develop new ways of working that take advantage of these new opportunities. For quicker progress on these problems, however, our best hope is to develop a more systematic theoretical and empirical foundation for understanding organizational processes. If we are to understand successful organizational practices, we must be able to recognize and represent the organizational practices we see. And to improve organizational practice in a particular situation, we must also be able to imagine alternative ways of accomplishing the same things. Finally, we need some way of judging which alternatives are likely to be useful or desirable in which situations.

This paper reports on the first five years of work in a project to address these problems by (1) developing methodologies and software tools for representing and codifying organizational processes at varying levels of abstraction and (2) collecting, organizing, and analyzing numerous examples of how different groups and companies perform similar functions. The result of this work is an on-line "process handbook" which can be used to help people: (1) redesign existing business processes, (2) invent new processes (especially those that take advantage of information technology), and (3) organize and share knowledge about organizational practices. We also expect this process handbook to be useful in automatically (or semiautomatically) generating software to support or analyze business processes, but that is not the focus of this paper (see Dellarocas, 1996, 1997a, 1997b).

The goal of compiling a complete handbook of business processes is, of course, a never-ending task. Our goal in this research project is more modest: to provide a "proof of concept" that limited versions of such a handbook are both technically feasible and managerially useful. Even though this project is not yet complete, the initial goal of demonstrating the basic technical feasibility of this approach has been achieved, and that is the primary focus of this paper. We have also conducted field tests that demonstrate the potential managerial usefulness of such handbooks and we include a description of one such test, as well.

## THE KEY INTELLECTUAL CHALLENGE:

# HOW TO REPRESENT ORGANIZATIONAL PROCESSES?

In order to develop a system that could be used in the ways listed above, the key theoretical challenge is to develop techniques for representing processes. Fortunately, the last several decades of research in computer science and other disciplines have resulted in a number of well-developed approaches to representing processes, such as flow charts and data-flow diagrams (e.g., Yourdon, 1989), state transition diagrams (e.g., Lewis & Papadimitriou, 1981; Winograd & Flores, 1986), Petri nets (e.g., Peterson, 1977; Holt, 1988; Singh & Rein, 1992), and goal-based models (e.g., Yu, 1992). These approaches have been used by many organizations to map their own specific processes, and some have used them to represent widely-used generic processes (e.g., Scheer, 1994; Maull, Childe, Bennett, Weaver, & Smart, 1995; Winograd & Flores, 1986; Carlson, 1979). For example, a number of consulting firms and other organizations have already developed "best practice" databases that include verbal descriptions, key concepts, and sometimes detailed process maps for a variety of generic processes such as logistics, marketing, and manufacturing (e.g., Peters, 1992, pp. 387-390; CIO Magazine, 1992). It is clear, therefore, that it is technically feasible to assemble a large set of process descriptions collected from many different organizations. It is also clear that such libraries of process descriptions can be useful to managers and consultants. The research question, then, is not whether it is possible to have a useful repository of knowledge about business processes. These databases already demonstrate that it is. Instead, the question is "How can we do better than these early databases?"

To answer this question, we have developed a new approach to analyzing and representing organizational processes that explicitly represents the similarities (and the differences) among a collection of related processes. Our representation exploits two sources of intellectual leverage: (1) notions of *specialization of processes* based on ideas about inheritance from object-oriented programming, and (2) concepts about *managingdependencies* from coordination theory.

### Specialization of processes

Most process mapping techniques analyze business processes using only one primary dimension: breaking a process into its different *parts*. Our representation adds a second dimension: differentiating a process into its different *types*. Figure 1 illustrates the difference between these two dimensions. In this figure, the generic activity called "Sell product" is broken apart into parts (or *subactivities*) like "Identify potential customers" and "Inform potential customers." The generic activity is also differentiated into types (or *specializations*) like "Sell by mail order" and "Sell in retail store".

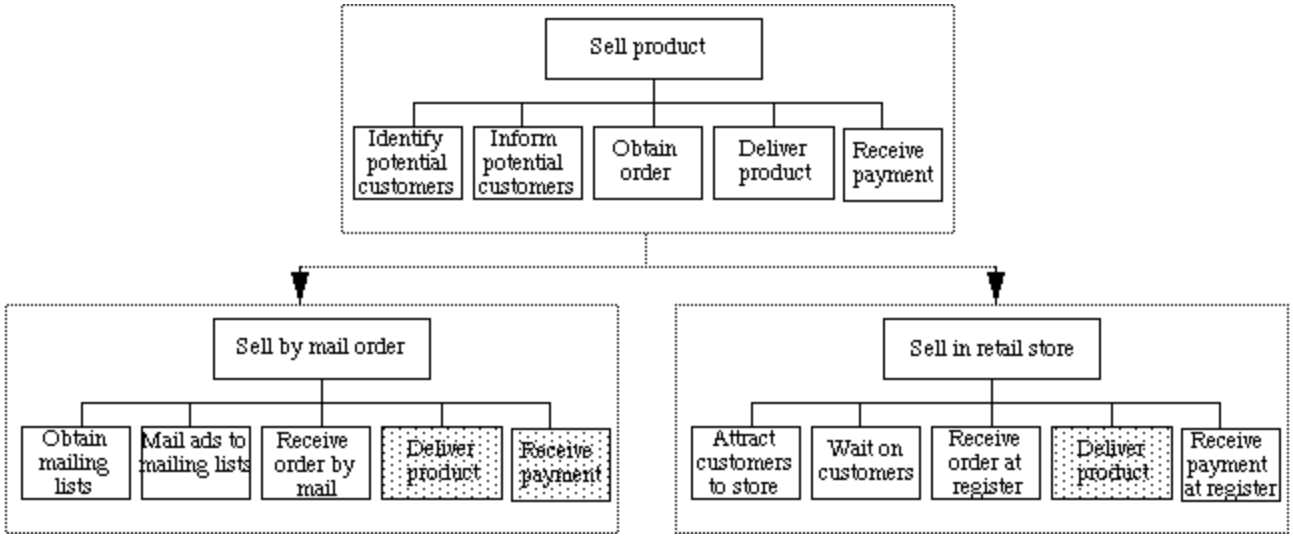


**Figure 1.** Sample representations of three different sales processes. "Sell by mail order" and "Sell in retail store", are specializations of the generic sales process "Sell something". Subactivities that are inherited without change are shaded.

---

As in object-oriented programming (e.g., Stefik & Bobrow, 1986; Wegner, 1987; Brachman & Levesque, 1985), the specialized processes automatically inherit properties of their more generic "parents", except where they explicitly add or change a property. For instance, in "Sell by mail order", the subactivities of "delivering a product" and "receiving payment" are inherited without modification, but "Identifying prospects" is replaced by the more specialized activity of "Obtaining mailing lists."

Using this approach, any number of activities can be arranged in a richly interconnected two-dimensional network. Each of the subactivities shown in Figure 1, for instance, can be further broken down into more detailed subactivities (e.g., "Type mailing list name into computer") or more specialized types (e.g., "Sell hamburgers at McDonald's retail restaurant #493") to any level desired. In general, we use the term "activity" for all business processes, including all their subparts and subtypes at all levels.

We have found the "process compass" shown in Figure 2 to be a useful way of summarizing the two dimensions. The vertical dimension represents the conventional way of analyzing processes: according to their different *parts*. The horizontal dimension is the novel one: analyzing processes according to their different *types*. From any activity in the Process Handbook, you can go in four different directions: (1) *down* to the different parts of the activity (its

"subactivities"), (2) *up* to the larger activities of which this one is a part (its "uses"), (3) *right* to the different types of this activity (its "specializations"), and (4) *left* to the different activities of which this one is a type (its "generalizations").

---


**Figure 2.**The "Process Compass" illustrates two dimensions for analyzing business processes. The vertical dimension distinguishes different parts of a process; the horizontal dimension distinguishes different types of a process.

---

*Comparison with object-oriented programming*

To readers familiar with conventional object-oriented programming techniques, it is worth commenting on the difference between our approach and conventional object-oriented programming. The difference is a subtle, but important, shift of perspective from specializing *objects* to specializing *processes* (see Stefik, 1981; Friedland, 1979; Thomsen, 1987; Madsen, Moller-Pedersen, & Nygard, 1993; Wyner & Lee, 1995; and other references in the section below on related work in computer science).

In a sense, this approach is a kind of "dual" of the traditional object-oriented approach. Traditional object-oriented programming includes a hierarchy of increasingly specialized *objects*, which may have associated with them *actions* (or "methods"). Our approach, by contrast, includes a hierarchy of increasingly specialized *actions* (or "processes") which may have associated with them *objects*. Loosely speaking, then, traditional object-oriented programming involves inheriting down a hierarchy of *nouns*; our approach involves inheriting down a hierarchy of *verbs*.

In a sense, of course, these two approaches are formally equivalent: anything that can be done in one could be done in the other. The two approaches can also, quite usefully, coexist in the same system. The process-oriented approach we are describing, however, appears to be particularly appropriate for the analysis and design of business processes.

*Bundles and trade-off tables*

In developing tools to support specialization, we have found it useful to combine specializations into what we call "bundles" of related alternatives. These bundles do not have a direct parallel in traditional object-oriented languages; however, they are comparable to "facets" in information science (Rowley, 1992). For instance, Figure 3 shows part of the specialization hierarchy for sales processes. In this example, one bundle of specializations for "Sell something" is related to *how* the sale is made: direct mail, retail storefront, or direct sales force. Another bundle of specializations has to do with *what* is being sold: beer, automotive components, financial services, etc.

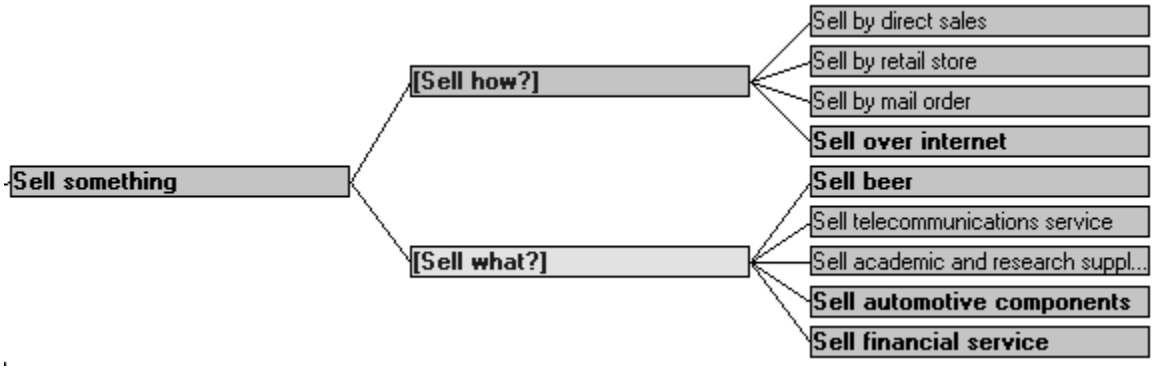


**Figure 3.**Summary display showing specializations of the activity "Sell something". Items in brackets (such as "[Sell how?]") are "bundles" which group together sets of related specializations. Items in bold have further specializations. (Note: The screen images used in this and subsequent figures were created with the software tools described below.)

Comparing alternative specializations is usually meaningful only *within* a bundle of related alternatives. For example, comparing "retail store front sales" to "direct mail sales" is sensible, but comparing "retail store front sales" to "selling automotive components" is not. Where there are related alternative specializations in a bundle, our handbook can include comparisons of the alternatives on multiple dimensions, thus making explicit the tradeoff between these dimensions. For example, Figure 4 shows a "tradeoff matrix" that compares alternatives in terms of their ratings on various criteria; different specializations are the rows and different characteristics are the columns. As in the Sibyl system (Lee & Lai, 1991), items in the cells of this matrix can be associated with detailed justifications for the various ratings. For very generic processes such as those shown here, the cells would usually contain rough qualitative comparisons (such as "High", "Medium", and "Low"); for specific

process examples, they may contain detailed quantitative performance metrics for time, cost, job satisfaction, or other factors. In some cases, these comparisons may be the result of systematic studies; in others, they may be simply rough guesses by knowledgeable managers or consultants (with appropriate indications of their preliminary nature); and, of course, in some cases, there may not be enough information to include any comparisons at all.

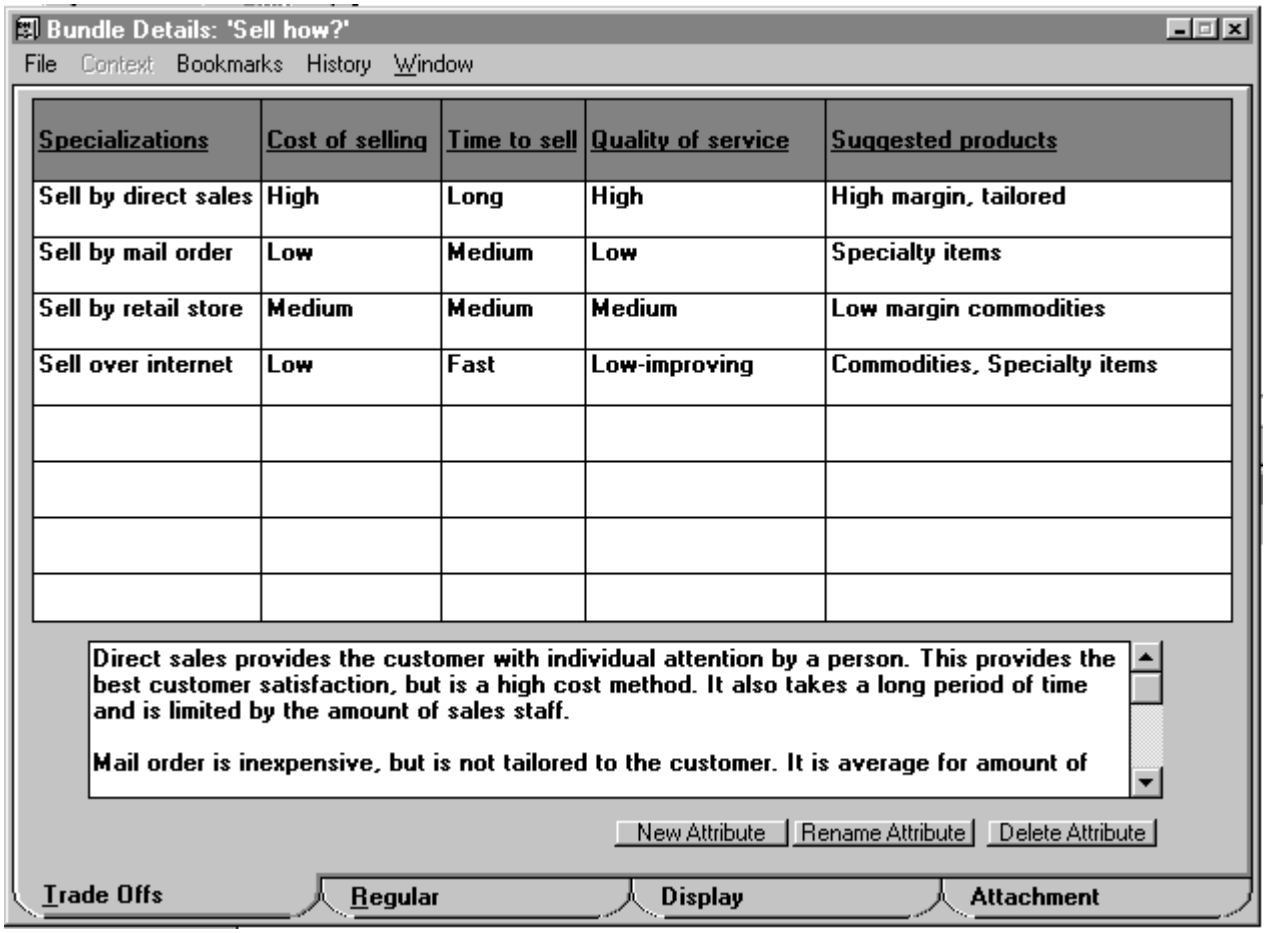


**Figure 4.** A tradeoff matrix showing typical advantages and disadvantages of different specializations for the generic sales process. (Note that the values in this version of the matrix are not intended to be definitive, merely suggestive.)

## Dependencies and coordination

The second key concept we are using is the notion from coordination theory (e.g., Malone & Crowston, 1994) that *coordination* can be defined as *managing dependencies among activities*. From this perspective, we can characterize different kinds of *dependencies* and the alternative *coordination processes* that

can manage them. Such coordination processes are both ubiquitous (i.e., the same mechanisms are found in many different processes) and variable (i.e., there are many different mechanisms that can be used to manage a particular dependency). Therefore, identifying dependencies and coordination mechanisms offers special leverage for redesigning processes. The power of analyzing processes in terms of dependencies and coordination mechanisms is greatly increased by access to a rich library of alternative coordination mechanisms for different kinds of dependencies. Therefore, a critical component of the Process Handbook is a library of generic coordination mechanisms.

Figure 5 suggests the beginnings of such an analysis (see Crowston, 1991; Zlotkin, 1995). The figure shows three basic kinds of dependencies: *flow, sharing,* and *fit*. These three types of dependencies arise from resources that are related to multiple activities. *Flow dependencies* arise whenever one activity produces a resource that is used by another activity. This kind of dependency occurs all the time in almost all processes and is the focus of most existing process mapping techniques (such as flow charts). *Sharing dependencies* occur whenever multiple activities all use the same resource. For example, this kind of dependency arises when two activities need to be done by the same person, when they need to use the same machine on a factory floor, or when they both use money from the same budget. Even though this kind of dependency between activities is usually omitted from flow charts, allocating shared resources is clearly a critical aspect of many management activities. Finally, *fit dependencies* arise when multiple activities collectively produce a single resource. For example, when several different engineers are designing different parts of a car (such as the engine, the transmission, and the body) there is a dependency between their activities that results from the fact that the pieces they are each designing need to fit together in the completed car.

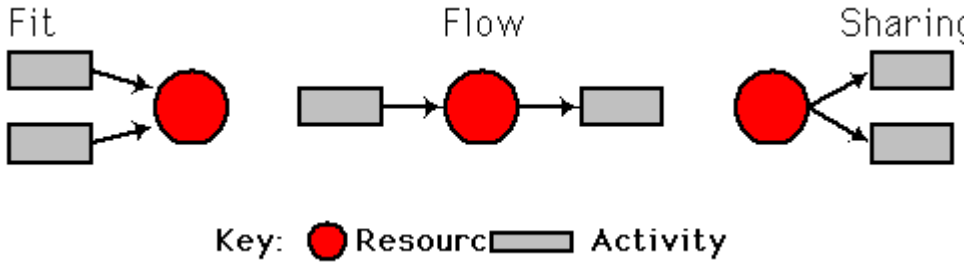


**Figure 5.** Three basic types of dependencies among activities (adapted from Zlotkin, 1995).

Table 1 extends this analysis by showing how the different kinds of dependencies can be associated with a set of alternative coordination processes for managing them. For example, the table shows that "sharing" dependencies (shared resource constraints) can be managed by a variety of coordination mechanisms such as "first come/first serve", priority order, budgets, managerial decision, and market-like bidding. If three job shop workers need to use the same machine, for instance, they could use a simple "first come/first serve" mechanism. Alternatively, they could use a form of budgeting with each worker having pre-assigned time slots, or a manager could explicitly decide what to do whenever two workers wanted to use the machine at the same time. In some cases, the owner might even want to sell time on the machine and the person willing to pay the most would get it. In this way, new processes can be generated by considering alternative coordination mechanisms for a given dependency.

| *Dependency* | *Examples of coordination mechanisms for managing dependency* |
|---|---|
| **Flow** | |
| Prerequisite ("right time") | Make to order vs. make to inventory ("pull" vs."push").<br><br>Place orders using "economic order quantity","Just In Time" (kanban system), or detailed advanced planning. |
| Accessibility ("right place") | Ship by various transportation modes or make at point of use |
| Usability ("right thing") | Use standards or ask individual users (e.g., by having customer agreeto purchase and/or by using participatory design) |
| **Sharing** | "First come/firstserve", priority order, budgets, managerial decision, market-like bidding |
| **Fit** | Boeing's total simulations. Microsoft's daily build |

**Table 1.** Examples of elementary dependencies between activities and alternative coordination mechanisms for managing them.

While the dependencies shown in Table 1 are certainly not the only ones possible, our current working hypothesis is that all other dependencies can be usefully analyzed as specializations or combinations of those shown in the table. Similarly, even though there are many other possible coordination processes, the table illustrates how a library of generic coordination processes can be organized according to the dependencies they manage.

*Specialization and decomposition of dependencies*

Some dependencies can be viewed as specializations of others. For instance, *task assignment* can be seen as a special case of sharing, where the "resource" being shared is the time of people who can do the tasks. This implies that the coordination mechanisms for sharing in general can be specialized to apply to task assignment. In other cases, some dependencies can be seen as being composed of others. For instance, *flow dependencies* can be viewed as a combination of three other kinds of dependencies: *prerequisite* constraints (an item must be produced before it can be used), *accessibility* constraints (an item that is produced must be made available for use), and *usability* constraints, (an item that is produced should be "usable" by the activity that uses it). Loosely speaking, managing these three dependencies amounts to having the *right thing* (usability), in the right place (accessibility), at the right time (prerequisite). Each of these different kinds of dependencies, in turn, may have different processes for managing it; for example, the prerequisite dependency might be managed by keeping an inventory of the resource or by making it to order when it is needed, while usability may be managed through a product design process.

## Related work in organization theory and design

In some respects, this work represents another step on what Sanchez (1993, p. 73) calls "the long and thorny way to an organizational taxonomy." Because our work draws heavily on the concept of specialization (and therefore classification), it is related to other taxonomies of organizations (e.g., Woodward, 1965; Thompson, 1967; Pugh, Hickson and Hinings, 1968; Mintzberg, 1979; Ulrich and McKelvey, 1990; Salancik & Leblebici, 1988). The main difference is that except for Salancik & Leblebici (1988), most work in this area has classified whole organizations (or parts of organizations). Instead, we classify processes. McKelvey (1982) argues that the study of organizations is at a "pre-Linnaean" stage, awaiting a more systematic taxonomy to enable further scientific progress. By focusing on processes, the perspective introduced here extends previous work and provides a significant new alternative in this important problem area.

For example, our work not only provides a framework for classification, but also a framework for identifying possible alternatives and improvements. Previously, Salancik and Leblebici (1988) introduced a grammatical approach to analyzing specific organizational processes that enabled the generation of new processes by the constrained rearrangement of component activities. Our representation extends this approach, adding specialization and inheritance of activities as well as explicit representation of various kinds of dependencies. Specialization enables us to generate new processes by using alternative sets of more primitive actions. Explicit representation of dependencies allows us to generate many possible coordination processes for managing these dependencies. For example,

Salancik and Leblebiciís alternative orderings can all be generated as alternative ways of coordinating the basic flow and other dependencies among the activities.

Our framework also emphasizes the importance of coordination in organizational design. Our concept of dependencies, for instance, elaborates on and refines the traditional concept of interdependence from organization theory (Thompson, 1967). As Thompson (1967) makes clear, interdependence between organizational subunits is a result of the way workflows are organized between them. Thompson identified three kinds of interdependence: pooled, sequential, and reciprocal. For each of these, he identified typical coordination strategies, such as standardization, planning, and mutual adjustment. As these concepts have been applied over the years, however, the concept of interdependence has come to describe relationships between organizational subunits. In a sense, therefore, our approach reasserts Thompsonís (1967) original insight by emphasizing that dependencies arise between activities in a process, not between departments *per se*. We extend Thompsonís (1967) work by identifying a much finer grained set of dependencies and a much richer set of coordination mechanisms for managing them.

We are able to explicitly relate dependencies and coordination mechanisms in this manner because our typology of dependencies is based on the pattern of use of common resources that creates the dependency, rather than on the topology of the relationship between the actors, as in Thompson's three categories. This approach makes it clearer which coordination mechanisms should be considered as alternatives, namely those that address the same kinds and uses of resources.

In representing processes computationally, our work is also similar to other computational organizational models (e.g., Cohen, March, & Olsen, 1972; Carley et al., 1992; Levitt, et al., 1994; Gasser & Majchrzak, 1994; Baligh, Burton, & Obel, 1990; Masuch & LaPotin, 1989). One major difference from most of this work, however, is that we focus on *organizing knowledge,* not on *simulating performance*. We can, of course, include simulation models and their results in the knowledge we organize, but our focus is on useful ways of organizing this knowledge, not on generating it.

For instance, Carley et al. (1992) developed Plural Soar, a simulation of a team of actors retrieving items from a warehouse. They used this simulation to study the effect of communications between actors and of individual memory on the performance of the group. In our system, the basic processes followed by the group could be stored and specialized to include or omit communication and memory. We could also include the performance of each variation as found from the simulation.

The Process Interchange Format (PIF), described below, is intended to simplify the task of translating process descriptions between a wide variety of such systems.

**Related work in computer science**

The idea of generic processes (or "scripts" or "plans") has a long history in the field of artificial intelligence (e.g., Schank & Abelson, 1977; Schank, 1982; Chandrasekaran, 1983; Clancey, 1983; Tenenberg, 1986; Bhandaru & Croft, 1990; Lefkowitz & Croft, 1990; Chandrasekaran, et al., 1992; Marques, et al., 1992). Of particular relevance to our work is the work on "skeletal plans" (Stefik, 1981; Friedland, 1979; Friedland & Iwakasi, 1985), where an abstract plan is successively elaborated (and "specialized") for a given task. The Process Handbook can also be viewed as a case-based reasoner (Kolodner, 1993) since many of the processes represented in the Handbook are case examples from specific organizations.

Unlike these AI systems, however, the Process Handbook uses both process specialization and dependencies with coordination mechanisms to generate and organize a large number of examples and generalizations about them. For example, unlike a conventional case-based reasoner with only a library of previous cases, the Process Handbook can also contain an extensive (human-generated) network of generic processes that summarize and organize the existing cases and that also help generate and evaluate new possibilities.

Outside the area of artificial intelligence, the notion of specializing processes has also been used occasionally in other parts of computer science. For example, a few programming languages (e.g., Thomsen, 1987; Madsen, Moller-Pedersen, & Nygard, 1993) include mechanisms for defining specialization hierarchies of processes and combining actions from different levels in various ways at run-time. However, even in the parts of computer science where this work has been done, the potential power of systematically inheriting patterns of activities, dependencies, and other properties though networks of increasingly specialized processes does not seem to be widely appreciated.

In recent years, the idea of explicitly representing the processes associated with connections between activities has begun to receive some attention (e.g., Stovsky and Weide, 1988). For example, several recent architectural description languages (ADLs) are used to describe software systems in terms of components and connectors, where both components and connectors are first-class entities (Allen and Garlan, 1994; Shaw et. al., 1995; Shaw and Garlan, 1996). Components are analogous to our activities, while connectors correspond to our coordination processes. However, in these ADLs connectors are implementation-level abstractions (such as a pipe, or a client/server protocol). In contrast, the

process handbook notion of dependencies also supports hierarchies of specification-level abstractions for interconnection relationships.

A key difference between our work and most previous work in all these areas of computer science comes from the difference in goals. The previous work in artificial intelligence and programming languages was primarily focused on building computer systems that, themselves, design or carry out processes. Our primary goal, on the other hand, is to build computer systems that help people design or carry out processes.

Because we have focused on *supporting* human decision-makersónot replacing them--there is no requirement that all our process descriptions be detailed or formalized enough to be executable by automated systems. Instead, it is up to the users of the Handbook to describe different processes at different levels of detail depending on their needs and the costs and benefits of going to more detailed levels. Therefore, unlike some of the well-known attempts to create comprehensive ontologies of actions (e.g. Lenat, 1995; Schank and Abelson, 1977), users of the Process Handbook do not have to wait for the resolution of difficult knowledge representation issues nor invest a large amount of effort in formalizing knowledge that is not immediately useful.

For domains in which the processes are formalized in enough detail, however, the Handbook can greatly facilitate the re-use of previously defined models such as simulations, workflow systems, transaction processing systems, or other software modules (e.g., Dellarocas, 1996, 1997a, 1997b).

## RESULTS

The combination of approaches described above should make it practical to store large numbers of processes, and, more importantly, enable users to generate a rich set of possible alternative processes. To test the feasibility of our approaches, we developed a series of prototype versions of a Process Handbook. The primary results of this work have been a set of *software tools* for viewing and manipulating process descriptions and a body of *information content* about business processes. In addition to these primary results, this section also includes brief descriptions of our *methodologies* for analyzing and organizing process descriptions and a *field test* of our approach.

### Software tools: The Process Handbook system

To date, the most visible product of our project is a set of software tools for storing and manipulating process descriptions. The core system manages the database of process descriptions and displays and edits selected entries. Our current system is implemented under the Microsoft Windows operating system

using Microsoftís Visual Basic programming language and numerous third-party modules for that environment (i.e., VBXs). The process descriptions are stored in a relational database (currently Microsoft Access) with an interface layer above the database that represents processes using the concepts described above (Ahmed, 1995; Bernstein, Dellarocas, Malone, & Quimby, 1995). This interface allows users to retrieve, view and edit process descriptions, including adding new subactivities and specializations.

The user interface includes: (1) *templates for describing activities*, including standard fields (like name, description, and author) and custom fields for specialized information about particular kinds of activities, (2) *links between activities*, including standard links (like generalizations, specializations, and subactivities), as well as arbitrary "navigational links" with which users can group activities in any way they want; and (3) *summary views of specializations and decompositions*, which allow direct manipulation of the database, including operations such as adding, changing, deleting, or moving entries.

The system also provides: (4) *automated support for inheritance*,so that changes in an activity are automatically made in all its specializations that have not over-ridden them, and (5) *automated support for dependencies*, so that users can specify the kind of dependency that exists between two or more activities and then search the space of possible coordination mechanisms for that dependency to identify a coordination mechanism (Elly, 1996).

With this last feature, users can easily switch back and forth between viewing the dependency or the coordination mechanism that manages the dependency (see Figure 6). By successively replacing dependencies with coordination mechanisms and activities with their specializations users can easily see many different views of the same process, from the most abstract to the most detailed.

---

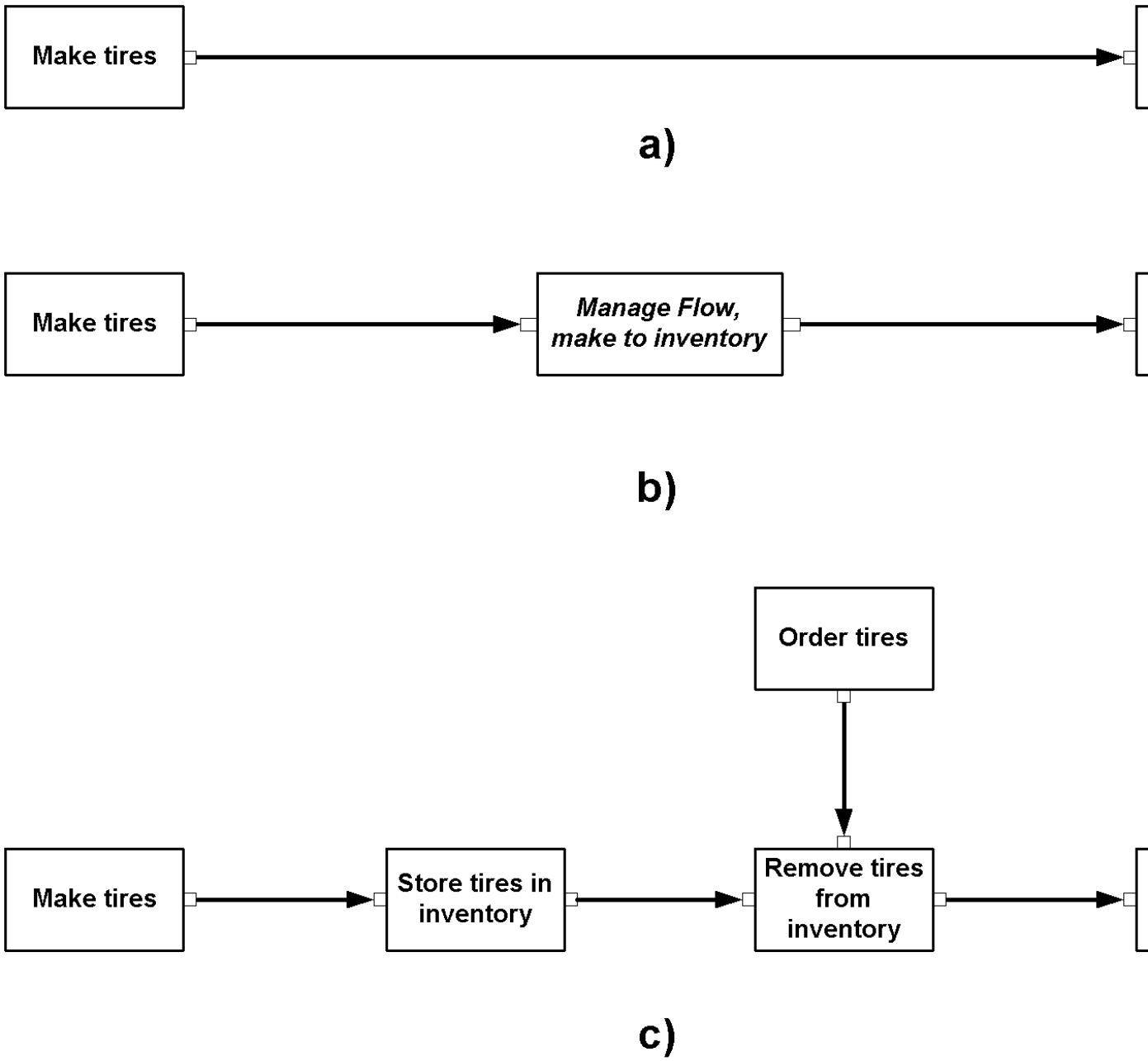


**Figure 6.** Alternative views of the same sample process. The first view (a) shows a "flow" dependency between two activities. The second view (b) shows the flow dependency replaced by the coordination process that manages it. The third view (c) shows the subactivities of the coordination process and the respective dependencies among them. Users can easily switch back and forth among these different views of the same process.

---

*Web interface*

We have also developed a World Wide Web interface to the system that allows users to view (but not to change) the contents of the Process Handbook from anywhere on the Internet. Using a standard Web browser, users can see information structured with templates, links, and inheritance, and they can contribute to on-line discussions about each of the activities.

*Process Interchange Format*

While we believe the tool described above has several unique advantages, there are many other process tools available for tasks such as flowcharting, simulation, workflow, and Computer-Aided Software Engineering (CASE). To increase the potential sources and uses for process descriptions in the handbook, we wanted to be able to move processes back and forth between these different tools. To help make this possibility more likely, we organized a working group, including people from our project and from several other university research groups and companies sponsoring our research. This group has developed a Process Interchange Format (PIF) for moving process descriptions between systems that use diverse representations (Lee et al, 1994; Lee et al, 1996). Via PIF, a process in one system (e.g. a process modeller) can be used by another (say, a simulator), whose result in turn can be used by yet another system. Each system uses as much as possible of the process descriptions and passes on information it cannot "understand" to other systems (Lee & Malone, 1990; Chan, 1995).

## Information content: The Process Handbook database

To test the feasibility of our approach it was critical to enter a significant number of process descriptions into the system. As Table 2 summarizes, the handbook currently contains over 3400 activities, some from specific organizations and some generic processes. This information content is the second major result of our work to date.

---

| *Kind of activity* | *Approx. no. of specific organi-zations repre-sented* | *Approx. no. of activities* | *Maximum no. of levels of speciali-zation* | *Maximum no. of levels of decom-position* | *Sample activity names* |
|---|---|---|---|---|---|
| **Examples from specific organizations** | | | | | |
| Manufacturing | 3 | 325 | 2 | 6 | Brew beer |
| Other "supply | 4 | 235 | 4 | 5 | Build walls |

| | | | | | |
|---|---|---|---|---|---|
| chain" processes | | | | | |
| Others | 30 | 60 | 2 | 2 | Select human resources |
| **Generic processes** | | | | | |
| Generic business processes | N/A | 70 | 3 | 4 | Sell something |
| Generic coordination processes | N/A | 100 | 7 | 2 | Manage accessibility by collocation |
| Other generic activities | N/A | 1165 | 4 | 9 | Acquire human resources |
| **Total** | 37 | 1955 | 7 | 9 | |

**Table 3.** Summary of current contents of the Process Handbook database (as of 10/1/96)

---

*Examples from specific organizations*

In addition to using secondary sources of data (such as published descriptions of innovative business practices), we have focused our primary data collection on the domain of "supply chain management" -- the process by which an organization (or group of organizations) manages the acquisition of inputs, the successive transformations of these inputs into products, and the distribution of these products to customers. For example, the handbook includes results from several MIT mastersí thesis studies of supply chain processes ranging from a Mexican beer factory to a university purchasing process (Geisler, 1995; Leavitt, 1995; Lyon, 1995; Ruelas Gossi, 1995). The entries also include a number of examples drawn from the "Interesting Organizations Database" collected from published sources and student projects as part of an MIT research initiative on "Inventing the Organizations of the 21st Century."

*Generic business processes*

To take advantage of inheritance and to help find useful process analogies, we need to integrate specific process examples into a more general framework. To develop such a framework of generic processes, we first reviewed generic business process models from a variety of published sources (e.g., Davenport, 1993). Based on this work, we defined the broadest *organizational* process in the

Process Handbook as "Produce something." This term is intended to include both manufacturing organizations (which produce products) and service organizations (which produce services). We intend that every activity that occurs in an organization should fit somewhere in one of the five subactivities of this all-encompassing process: (1) design, (2) purchasing and inbound logistics, (3) production, (4) sales and outbound logistics, and (5) general management and administrative functions. Drawing on our general knowledge of business and a variety of published sources, including textbooks in marketing (Kotler, 1997) and product design (Ulrich & Eppinger, 1995), we have developed several levels of more detailed subactivities for these generic business activities.

However, the Process Handbook does not force a single perspective on these activities. For example, several of the generic business process models we reviewed are now included in the handbook as alternative specializations of "Produce something." These different models provide different *views* of how a business can be decomposed into subactivities. When several different specializations of an activity all include the same lower level subactivities, but group them in different ways we define the different specializations as alternative "views". Many such views are possible, and they are all functionally equivalent, so it would not make sense to claim that any particular set of generic business processes is definitive or intrinsically superior. Instead, users can pick the views they find most useful or appealing.

*Other generic activities*

In addition to the high-level generic business processes and generic coordination mechanisms described above, many other kinds of activities occur as basic building blocks of business processes. For example, activities like making a decision or approving an application are parts of many organizational processes. In order to take advantage of process inheritance and maximize the generativity of our framework, all activities need to be placed somewhere in the specialization hierarchy.

We have explored several alternatives for how to organize the specialization hierarchy that makes this possible. The most promising approach we have found so far (which we currently use in the handbook) is illustrated in Figure 7. The basic idea is to create a high-level framework of a small number of very generic activities, and then to classify all other activities as specializations of these high-level activities.

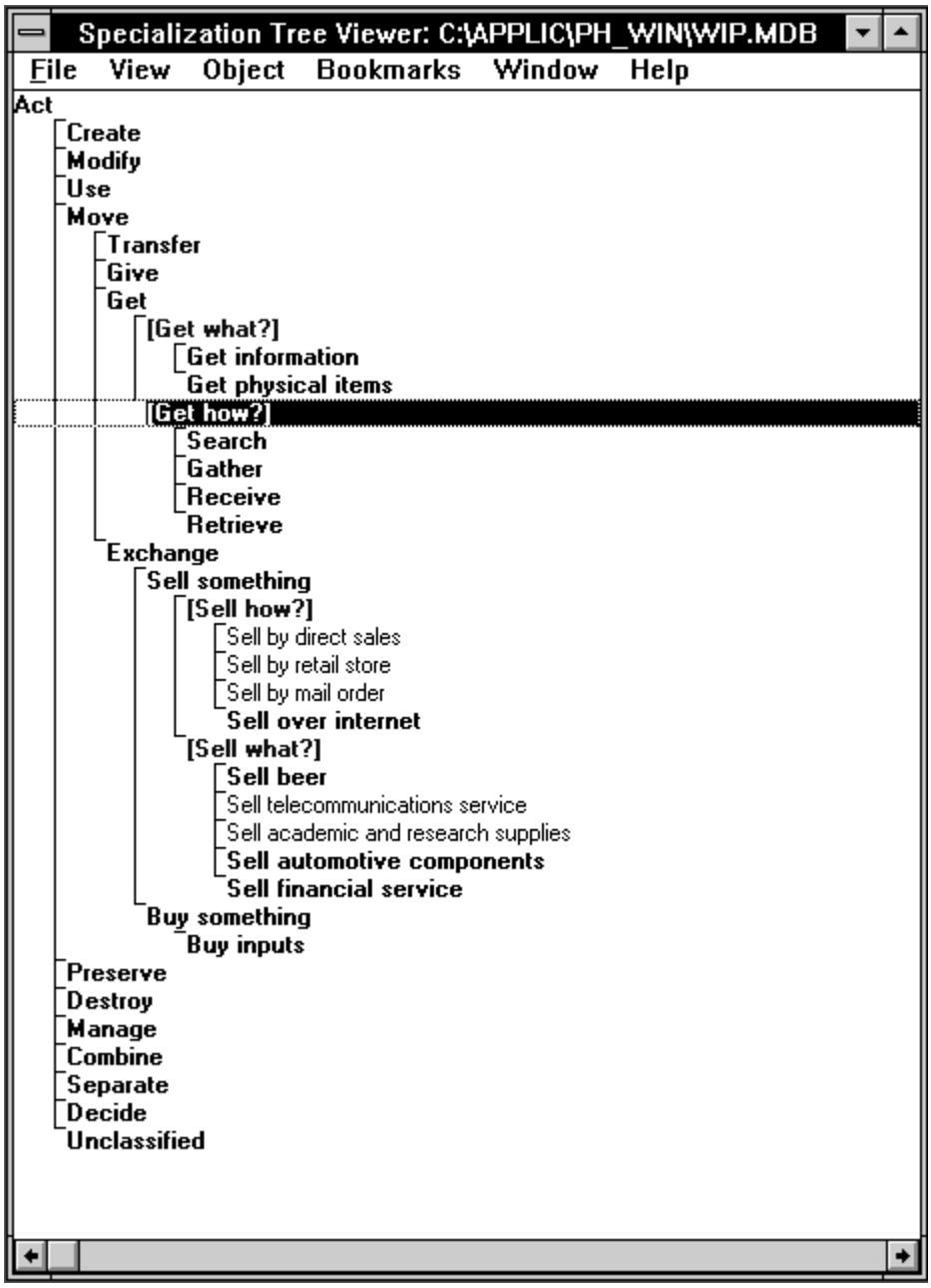


**Figure 7.** An outline view of the first two levels of the specialization hierarchy and selected further specializations of the generic activity "Move" (as of 11/1/96).

In the current version of this taxonomy, the top level consists of very general activities like Create, Destroy, Modify, and Preserve. These most general processes can occur for any kind of object. As the table illustrates, these generic processes are further specialized down to the lowest level of activity in the handbook. We have found it useful in many cases to group specializations into bundles based on questions about who, what, where, why, when, and how. For

example, the bundles under the generic "Get" activity, include "Get what?" and "Get how?" As with the other areas of the Process Handbook, the further development of this part of the process taxonomy is an active part of our ongoing research. The taxonomy we have developed so far demonstrates the basic feasibility of organizing large numbers of activities in a unified specialization hierarchy.

## Methodologies

For this approach to be feasible for large-scale use, we need to be able to systematically analyze processes and integrate them into the Process Handbook. In addition to developing methods for analyzing processes (with or without the Process Handbook repository), we are also refining methods for editing and integrating information about processes into the handbook database. For instance, a "top down" approach to analyzing a new process for the handbook is to start with similar examples already in the handbook, create a new specialization, and then modify the specialization as needed to describe the new process. An alternative "bottom up" approach is to start by entering a description of the new process and then connecting it to existing processes in the handbook that are generalizations of the whole process or its subactivities. In the course of adding these new specializations to existing processes, the existing processes may be modified to include generalizations of elements in the new processes.

In many cases, we believe the best approach is a combination of both these approaches: working both top-down and bottom-up to successively refine both old and new process descriptions and maximizing the insights along the way. Our experiences with these methodologies are now being formalized (e.g., Crowston and Osborn, 1996; Pentland, et al., 1994) and integrated into teaching materials.

## Field testing the Process Handbook: A case study

In a sense, each new process description entered into the handbook is a field test of the framework, because it raises the question: can this process be adequately represented? But the more important question is: what can we get back from the handbook? What kinds of activities can this representation support? To answer this question, we have begun to field test the handbook in real organizations that are engaged in process improvement efforts. While not in any sense controlled experiments, these field studies provide concrete illustrations of the potential managerial usefulness of the Process Handbook concepts. One such study is summarized here (see Herman et al., 1997; and Roth, 1997 for additional details). This study was done in collaboration with one of our corporate research sponsors, the AT Kearney consulting firm, and one of their clients which we call "Firm A" to preserve the client's anonymity.

Firm A was experiencing increasing problems with their hiring process. They were growing rapidly in a tightening labor market, and they had a culture of independent, competitive business units. Together, these factors led to increases in the time and cost to hire people and to increasingly frequent instances of business units "hoarding" candidates or bidding against each other for the same candidate.

In an effort to improve their hiring process, the organization had invested a great deal of time and energy into "as is" process analysis using conventional techniques such as flowcharting. But they also wanted some way to come up with highly innovative ideas about how to improve their process. In this spirit, they agreed to participate in a field test of the Process Handbook system and concepts. A study team of about 8 people was formed consisting of members from MIT, AT Kearney, and Firm A.

The team's first step was simply to see how the hiring process was represented in the Process Handbook. Several of the steps in the Handbook activity called "Hire human resources" were similar to those already identified by the "as is" analysis (e.g., identify need, determine source, select, and make offer). One immediate insight, however, resulted from the fact that the Process Handbook representation of hiring included a step of "pay employee" which had not been included in the "as is" analysis. Even though they hadn't previously thought of it in this way, the team members from Firm A found it surprising and useful to realize that the employee receiving a first paycheck is, in a sense, the logical culmination of the hiring process. Receiving a (correct) paycheck, for instance, confirms that the hiring information has been entered correctly in the relevant administrative systems.

*Using the concepts of specialization*

To generate further insights and alternatives, the team looked in the Process Handbook at specializations of the overall hiring process and then at the specializations of each of its subactivities. In terms of the process compass mentioned above, the team looked first to the right, then down and to the right. In doing so, they came across examples such as Marriott Hotels, where an automated telephone system asks job candidates a series of questions about their qualifications and salary requirements. At the end of the call, callers are immediately told if they're qualified for the position and invited to schedule an interview through the system's automated scheduling feature. Although most appropriate for lower level personnel, this example was very thought provoking for the project team.

The team found numerous other similarly intriguing examples in the handbook. For example, they found descriptions of (1) BMW using a simulated assembly

line to help select assembly line workers, (2) Whirlpool having a corporate-wide "human capital war room" with databases of projected skill needs and capacities, and (3) Doubletree which seeks to systematically identify dimensions of employee success in their organization and then hire candidates with similar traits.

This use of the Process Handbook is similar to the traditional "benchmarking" or "best practice" approach of learning from other examples of the same process. Even here, however, the use of specialization in the handbook allows much richer ways of indexing large numbers of examples than any other "best practices" database of which we are aware.

In an effort to expand their horizons even further, the team's next step was to look in the handbook for more distant analogies (or "cousins") of the hiring process. That is, they looked first at generalizations ("ancestors") of the hiring process and then at other specializations ("descendants") of these generalizations. (In terms of the process compass, they moved left and then right again.)

For example, "hiring" is classified in the handbook as a specialization of "buying", so a handbook user who looks at the generalizations of "hiring" will encounter "buying". In retrospect, this connection may seem obvious (hiring is a form of buying someone's time), but this analogy had not been obvious to the project team, and it proved to be a very stimulating source of insights. In exploring other specializations of buying, for instance, the team encountered examples like (1) Motorola's extensive quality audits and rating systems for their suppliers, (2) Acer's different sourcing strategies for different kinds of materials, and (3) General Electric's Internet-based system through which purchasing agents can find and compare suppliers. Each of these examples stimulated specific ideas about possible improvements in the hiring process for Firm A: (1) quality ratings for recruiters, (2) creating different hiring processes for different kinds of positions, and (3) identifying candidates using the Internet, respectively.

*Using the concepts of coordination*

After exploring a number of such distant analogies, the team then began to systematically explore and compare many different possible combinations of specializations and coordination processes for hiring. One of the most interesting insights from this part of the process came from focusing on the shared resource dependency for recruiter time. Firm A used a variety of internal and external recruiters, and the time of these recruiters had to be somehow shared across all the positions being filled at any given time. The coordination process Firm A currently used for managing this dependency was to have recruiting managers for each business unit assign each new search to a specific recruiter.

When analyzing this process from a coordination point of view, the team quickly identified a variety of other possible ways to manage this dependency, including all the coordination processes listed for sharing dependencies in Table 1. The team was particularly intrigued by the idea of using market-like bidding systems for this purpose. In one scenario the team developed, for instance, recruiters would "bid" on the opportunity to fill a new position by specifying how long they estimated it would take them to fill the position. Later, when the position had actually been filled, the recruiter's fee would be adjusted for significant over- or under-performance relative to the original bid.

One compelling advantage of this scheme is that it could more easily exploit information that is often ignored completely in the current system. For instance, a recruiter who had just filled one position for a C++ programmer but who knew that 3 other highly qualified candidates identified in the same search were still available, could take this information into account in making a low bid on a new search for a C++ programmer in another business unit.

Our project ended before Firm A had implemented any of the ideas generated in this phase of the project, and no quantitative evaluation of the idea-generating phase of the project was done. However, in the meeting where the final project results were presented, the executive vice-president of human resources in Firm A eloquently articulated our aspirations in the project by saying that he felt he had "passed through a doorway where all sorts of things he had never imagined before now seemed possible."

## DISCUSSION

This case illustrates a number of advantages of using a specialization hierarchy in combination with the explicit representation of coordination and dependencies. First, this field test showed that specialization can substantially reduce the amount of work necessary to analyze a new process. By simply identifying a process as a "hiring process", for example, a great deal of information can be automatically inherited. Then, only the changes that matter for the purpose at hand need to be explicitly entered. This helps support a rapid assessment of the basic features of a process, rather than laborious detailing (what Hammer and Champy, 1993, refer to as "analysis paralysis"). For example in the field test, the team chose to ignore nearly all of the "as is" analysis that had previously been done by Firm A and focus on a very simple, abstract view of the hiring process and its first level subactivities. This level of detail, alone, was sufficient to generate all the insights described above.

Second, the specialization hierarchy provided a powerful framework for generating new process ideas. For example, some of today's "best practice" databases support cross-fertilization across industries within the same business

function, but we do not know of any others that would support the kind of cross-fertilization across business functions (from purchasing to human resources) described above.

Since coordination processes are often those most susceptible to being changed by information technology, a particularly important use of this approach is to use generic knowledge about alternative coordination mechanisms to generate new process ideas. For instance, the ideas about using bidding to allocate recruiter time were stimulated by very generic knowledge about coordination, and would presumably be more feasible because of the cheaper communication made possible by information technologies (see Crowston, 1997, for other similar examples).

Another feature of our approach that makes it particularly useful for generating new process ideas is that we focus attention on processes as distinct entities that can be described independently of organizational structures or the roles of particular people or groups. This "process-oriented" approach to business seems particularly useful, in (a) identifying new ways of doing old tasks, even if the new ways involve very different actors and (b) managing connected processes that span organizational boundaries: either across groups in a single firm or across firms in "networked" and "virtual" organizations.

In addition to these advantages, our process-oriented approach has limitations, too. For instance, any static process representation can give the impression that the process is more stable and routine than most business processes actually are. In contrast to most other process representations, however, our approach helps us explicitly deal with this issue by representing the stable--or typical--aspects of a process at the generic level and then also representing as many specialized variations as is useful.

Another risk of having libraries of explicit process representations like ours is that people will interpret them too rigidly. While it is sometimes appropriate to collect prescriptive rules or procedures in a handbook like ours, we think that in most situations a process handbook will be most useful as a *resource* to help people figure out what to do, rather than as a *prescription* of what they should do.

*The editorial challenge*

One of the most important ways in which our approach differs from many other computational approaches to similar problems is that we do not rely primarily on intelligent computer systems to analyze, reason about, or simulate business processes. Instead, we place substantial importance on the role of intelligent human "editors" to select, refine, and structure the knowledge represented in the handbook. This approach has both strengths and weaknesses.

On the one hand, it allows us to take advantage of human abilities to analyze, organize, and communicate knowledge in ways that go far beyond the capabilities of today's computers. For example, the task of developing good generic models for the marketing and sales process is similar, in many ways, to writing a good textbook or developing comprehensive theories about marketing and sales. Human abilities to do tasks like these will almost certainly exceed those of computers for the foreseeable future.

On the other hand, relying on human effort in this way means that the success of our approach depends significantly on the quality and amount of human intelligence applied to the problem of generating and organizing knowledge in the system. For example, a complex and confusing network of poorly organized process categories may be even worse than no categories at all.

In general, as process descriptions are added to the handbook, we will face a problem that is analogous to that faced by researchers in many fields: how to insure that results cumulate in a meaningful way. Since we foresee a wide variety of potential users and contributors, it would be unrealistic to expect equal rigor from all of them. Rather than attempting to enforce uniform standards, we plan to allow a wide variety of data from diverse sources, but to require that the specific sources, methods, and significance of that data be described in enough detail to allow users of the handbook to judge whether it is valid and reliable enough for their own purposes. In this respect, the Handbook has an advantage over more formal approaches because it allows many alternatives to co-exist in the system. At the same time, this openness contributes to the editorial problem of insuring that the entries are consistently and usefully classified. We believe that adopting solutions analogous to those that have already been found successful in other domains is a promising approach. For example, we have found it useful to think about roles like authors, editors, and reviewers for groups of entries in the Process Handbook.

It is also encouraging to note that the specialization structure of the handbook provides a potentially powerful advantage that has not been widely available to any knowledge generating communities before: Well-organized and accurate process knowledge at the "left" of the specialization network is automatically inherited throughout the other parts of the network where it applies. In this sense, then, the system amplifies the effort of intelligent humans by automatically linking their work to a variety of contexts where it may be useful.

## CONCLUSION

There is, of course, much more work to be done to develop and test the ideas described here. For example, better tools for process analysis and editing need to be created, more information content needs to be added to the Process

Handbook, and systematic tests of how the ideas can be applied in different kinds of situations need to be performed. However, we believe that our work so far has demonstrated the basic feasibility and contribution of the approach and its potential for significant further progress. We hope, for example, that this research will provide a set of intellectual tools and an extensive database to help people learn about organizations, invent new kinds of organizations, and improve existing processes. Perhaps most importantly, we hope this research will help us understand the possibilities for creating new kinds of organizations that are not only more effective, but also, more fulfilling for their members.

## ACKNOWLEDGMENTS

Parts of this paper appeared previously in Malone, Crowston, Lee, and Pentland (1993).

This work was supported, in part, by the National Science Foundation (Grant Nos. IRI-8903034, IRI-9224093, and DMI-9628949) and the Defense Advanced Research Projects Agency (DARPA). It was also supported by the following corporate sponsors: British Telecom, Daimler Benz, Digital Equipment Corporation, Electronic Data Systems (EDS), Fuji Xerox, Matsushita, National Westminster Bank, Statoil, Telia, Union Bank of Switzerland, Unilever, and other sponsors of the MIT Center for Coordination Science and the MIT Initiative on "Inventing the Organizations of the 21st Century". The software described in this paper is the subject of pending patent applications by MIT.

We would like to thank Marc Gerstein, Fred Luconi, and Gilad Zlotkin for their long-term contributions to many aspects of this project. We would like to thank John Gerhart for his significant early contributions to the content of the database and Martha Broad, Bob Halperin, Ed Heresniak, and Roanne Neuwirth for their contributions to the management of the project. We would also like to specifically thank the following students for their contributions to the development of the software tools described here: Erfan Ahmed, Frank Chan, Yassir Elley, Umar Farooq, Phil Grabner, Naved Khan, Vuong Nguyen, Greg Pal, Narasimha Rao, and Calvin Yuen. In addition, we would like to thank the dozens of students and others who contributed content to the database or who used the concepts developed in this project to analyze business processes. In particular, we would like to thank the following students whose work is specifically included in the current database: Gary Cheng, Martha Geisler, Paul Gutwald, Clarissa Hidalgo, Jeff Huang, Wilder Leavitt, William Lyon, Alejandro Ruelas Gossi, and Jin Xia. Finally, we would like to thank the members of the Process Handbook advisory board: Michael Cohen, John McDermott, and the late Gerald Salancik.

## REFERENCES


Allen, R. & Garlan, G. (1994). Formalizing Architectural Connection. In Proceedings of the 16th International Conference on Software Engineering, pages 71-80, Sorrento, Italy, March 1994.

Ahmed, Erfanuddin. *A Data Abstraction with Inheritance in the Process Handbook.* Unpublished M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1995.

Baligh, H. H., Burton, R. M. and Obel, B. (1990). Devising expert systems in organization theory: The Organizational Consultant. In M. Masuch (Ed.), *Organization, Management, and Expert Systems* (pp. 35-57). Berlin: Walter de Gruyter.

Bernstein, A., Dellarocas, C., Malone, T. W., & Quimby, J. (1995). Software tools for a Process Handbook. *IEEE Bulletin on Data Engineering*, *18*, 1 (March), pp.41-47.

Bhandaru, N. and Croft, W. B. An architecture for supporting goal-based cooperative work. In *Multi-User Interfaces and Applications,* S. Gibbs and A. A. Verrijin-Stuart (Ed.), Elsevier (North Holland), Amsterdam, 1990, pp. 337-354.

Brachman, R. J. and Levesque, H. J. (Eds.). (1985). *Readings in Knowledge Representation* . Los Altos, CA: Morgan Kaufmann.

Carley, K.., Kjaer-Hansen, J., Newell, A. and Prietula, M. (1992). Plural-Soar: Capabilities and coordination of multiple agents. In M. Masuch and M. Warglien (Eds.) *Artificial Intelligence in organization and management theory: Models of distributed intelligence* (pp. 87-118). New York: Elsevier Science.

Carlson, W. M. (1979). Business Information Analysis and Integration Technique (BIAIT) ó The new horizon. *Database*, *Spring*, 3-9.

Carter, D. E. and Baker, B. S. (1991). *Concurrent Engineering: The Product Development Environment for the 1990's*. Reading, MA: Addison-Wesley.

Chan, Frank Y. *The Round Trip Problem: A Solution for the Process Handbook.* Unpublished M.S. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1995.

Chandrasekaran, B. (1983). Towards a taxonomy of problem solving types. *AI Magazine*, *4*(1), 9-17.

Chandrasekaran, B., Johnson, T. R. and Smith, J. W. (1992). Task-structure analysis for knowledge modeling. *Communications of the ACM*, *35*(9), 124-137.

CIO Magazine (1992). Back Support for Benchmarkers, *CIO*, June 1, 1992, p. 16. (Note: This article gives a brief description of the activities of the International Benchmarking Clearinghouse. More detail is available from the following site on the World Wide Web: http://www.apqc.org/apqchome/apqchome.htm.)

Clancey, W. J. (1983). The epistemology of a rule-based expert system ó A framework for explanation. *Artificial Intelligence*, *20*(3), 215-251.

Cohen, M., March J. G., & Olsen J.P. (1972). A garbage can model of organizational choice. *Administrative Science Quarterly* 17:1.

Crowston, K. (1991). *Towards a Coordination Cookbook: Recipes for Multi-Agent Action*. Ph.D. Dissertation, MIT Sloan School of Management, Cambridge, MA.

Crowston, K. (1997). A coordination theory approach to organizational process design. *Organization Science*, *8* (2), 157-175.

Crowston, K. and Osborn, C. (1996). A coordination theory approach to process documentation and redesign. MIT Center for Coordination Science Working Paper, Massachusetts Institute of Technology, August.

Davenport, T. (1993). *Process Innovation: Reengineering Work through Information Technology.* Boston, MA: Harvard Business School Press.

Dellarocas, C. (1996). *A Coordination Perspective on Software Architecture: Towards a Design Handbook for Integrating Software Components*. Ph.D. Dissertation, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA.

Dellarocas, C. (1997a). Towards A Design Handbook for Integrating Software Components. *Proceedings of the 5th International Symposium on Assessment of Software Tools* (SAST'97), Pittsburgh, PA, June 2-5, 1997, pages 3-13.

Dellarocas, C. (1997b). The SYNTHESIS Environment for Component-Based Software Development. *Proceedings of the 8th International Workshop on Software Technology and Engineering Practice* (STEP'97), London, UK, July 14-18, 1997.

Dellarocas, C., Lee, J., Malone, T. W., Crowston, K., & Pentland, B. Using a process handbook to design organizational processes. *Proceedings of the AAAI '94 Stanford Spring Symposium on Computational Organization Design*., Stanford, CA, 1994.

Elley, Yassir (1996) *A Flexible Process Editor for the Process Handbook*. Master's Thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA.

Friedland, P. E. (1979) Knowledge-based Experiment Design in Molecular Genetics. Technical Rep. No. 79-771, Computer Science Dept., Stanford University (Doctoral Dissertation).

Friedland, P. E., & Iwasaki, Y. (1985). The concept and implementation of skeletal plans. *Journal of Automated Reasoning, 1* (2).

Gasser, L. & Majchrzak, A. (1994). ACTION Integrates Manufacturing Strategy, Design, and Planning. In P. Kidd and W. Karwowski (Eds.), *Ergonomics of Hybrid Automated Systems IV*, IOS Press, Netherlands.

Geisler, Martha A. *The Evolving Health Care Delivery Systems: Applying the Process Handbook Methodology to Gain a Vision of the Future*. Unpublished M.S. thesis, MIT Sloan School of Management, Cambridge, MA, May 1995.

Hammer, M. and Champy, J. (1993). *Reengineering the Corporation.* New York: Harper Business.

Herman, G., Klein, M., Malone, T. W., and O'Donnell, E. (1997) "A Template Based Methodology for Process Redesign", MIT Center for Coordination Science Unpublished working paper, October 1997.

Holt, A. W. (1988). Diplans: A new language for the study and implementation of coordination. *ACM Transactions on Office Information Systems*, *6*(2), 109-125.

Kolodner, J. (1993) *Case-based Reasoning*. San Mateo, CA: Morgan Kaufmann.

Kotler, Philip (1997). *Marketing management* (9th edition). New Jersey: Prentice Hall.

Leavitt, Wilder. *Health Care Delivery Systems: Using the MIT CCS Handbook to Create Organizations for the 21st Century*. Unpublished M.S. thesis, MIT Sloan School of Management, Cambridge, MA, May 1995.

Lee, J. & K.-Y. Lai. (1991). What's in design rationale? *Human-Computer Interaction*, *6*(3-4), 251-280.

Lee, J. and Malone, T. W. (1990). Partially shared views: A scheme for communicating among groups that use different type hierarchies. *ACM Transactions on Information Systems* , 8(1), 1-26.

Lee, J., G. Yost, and the PIF Working Group. (1994). *The PIF Process Interchange Format and Framework*. MIT CCS Working Report #180.

Lee, Jintae, Micheal Grunninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost and other members of the PIF Working Group (1996), The PIF Process Interchange Format and Framework Version 1.1, *Proceedings of Workshop on Ontological Engineering*, ECAI '96. Budapest, Hungary. (Also available as MIT Center for Coordination Science, Working Paper #194, 1996; and at the following World Wide Web site: http://ccs.mit.edu/pif.)

Lefkowitz, L. S., & Croft, W. B. Interactive planning for knowledge-based task management. Technical Report, Collaborative Systems Laboratory, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, May 1990.

Lenat, D. B. "CYC: A Large-Scale Investment in Knowledge Infrastructure." Communications of the ACM 38, no. 11 (November 1995).

Levitt, R. E., Cohen, G., Kunz, J.C., Nass, C.I., Christiansen, T., and Jin, Y. (1994) The Virtual Design Team: Simulating how organizations structure and information processing tools affect team performance. In Carley, K.M. and Prietula, M.J. (Eds.) *Computational Organization Theory*, Erlbaum: Hillsdale, N.J.

Lewis, H. R. and Papadimitriou, C. H. (1981). *Elements of the Theory of Computation*. New York: Prentice-Hall.

Lyon, William K. *The Process Handbook Supply Chain Reengineering*. Unpublished M.S. thesis, MIT Sloan School of Management, Cambridge, MA, May 1995.

Madsen, O. L., Moller-Pedersen, B., & Nygaard, K. (1993) *Object-Oriented Programming in the Beta Programming Language,* Reading, MA: Addison-Wesley.

Malone, T. W. and Crowston, K. (1994). The interdisciplinary study of coordination, *ACM Computing Surveys*, *26* (1), 87-119.

Malone, T. W., Crowston, K., Lee, J. and Pentland, B. (1993). Tools for inventing organizations: Toward a handbook of organizational processes. In *Proceedings of the 2nd IEEE Workshop on Enabling Technologies Infrastructure for Collaborative Enterprises*. Morgantown, WV, April 20-22.

Marques, D., Dallemagne, G., Klinker, G., McDermott, J. and Tung, D. (1992). Easy programming: Empowering people to build their own applications. *IEEE Expert*, *7,* 3 (June), 16-29.

Masuch, M., & LaPotin, P. (1989). Beyond Garbage Cans: An AI Model of Organizational Choice. *Administrative Science Quarterly*, 34: 38-67.

Maull, R., Childe, S., Bennett, J., Weaver, A., and Smart, A. (1995). Different types of manufacturing processes and IDEF0 models describing standard business processes. Working paper WP/GR/J95010-6, School of Computing, University of Plymouth, Plymouth, Devon, UK.

McKelvey, B. (1982). *Organizational SystematicsóTaxonomy, Evolution, Classification*. Berkeley: University of California Press.

Mintzberg, H. (1979). *The Structuring of Organizations*. Englewood Cliffs, NJ: Prentice-Hall.

Pentland, B. T. Osborne, C., Wyner, G., & Luconi, F. (1994). Useful descriptions of organizational processes: Collecting data for the Process Handbook. Unpublished working paper, Center for Coordination Science, Massachusetts Institute of Technology, Cambridge, MA.

Peters, T. *Liberation Management.* New York: Knopf, 1992.

Peterson, J. L. (1977). Petri nets. *ACM Computing Surveys*, *9*(3), 223-252.

Pugh, D.S., Hickson, D.J. and Hinings, C.R. (1968) An empirical taxonomy of work organizations, Administrative Science Quarterly, 14: 115-126.

Roth, G. (1997) Uniting theory and practice: An illustrative case for bridging knowledge and action. Unpublished working paper, MIT Initiative on Inventing the Organizations of the 21st Century.

Rowley, J. (1992). *Organizing Knowlege* (2nd ed.). Brookfield, VT: Ashgate.

Ruelas Gossi, Alejandro. *Inventing Organizations for the 21st Century in Mexico: Supply Chain Management in a Brewery*. Unpublished M.S. thesis, MIT Sloan School of Management, Cambridge, MA, May 1995.

Salancik, G. R. and Leblebici, H. (1988). Variety and form in organizing transactions: A generative grammer of organizations. In N. DiTomaso and S. B. Bacharach (Ed.), *Research in the Sociology of Organizations*. Greenwich, CT: JAI Press.

Sanchez, Julio C. (1993) The Long and Thorny Way to an Organizational Taxonomy, Organization Studies, 14(1): 73-92.

Schank, R. C. (1982). *Dynamic Memory: A theory of reminding and learning in computers and people*. New York: Cambridge University Press.

Schank, R. C. and Abelson, R. P. (1977). *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge*. Hillsdale, NJ: Lawerence Erlbaum Associates.

Scheer, A.-W. (1994). *Business Process Reengineering: Reference Models for Industrial Enterprises.* (2nd ed). New York: Springer-Verlag.

Shaw, M. et. al. (1995). Abstractions for Software Architecture and Tools to Support Them. IEEE Transactions on Software Engineering, pages 314-335, April 1995.

Shaw, M. & Garlan, D. (1996). Software Architecture: Perspectives on An Emerging Discipline. Prentice-Hall, Inc., Upper Saddle River, NJ.

Singh, B. and Rein, G. L. (1992). *Role Interaction Nets (RIN): A process description formalism* (Technical Report No. CT-083-92). Austin, TX: MCC.

Stefik, M. (1981) Planning with constraints (MOLGEN: Part 1). *Artificial Intelligence*, *16*(2), 111-139

Stefik, M. and Bobrow, D. G. (1986). Object-oriented programming: Themes and variations. *AI Magazine*, *6,* 4 (Winter), 40-62.

Stovsky, M.P. & Weide, B. W. (1988). Building Interprocess Communication Models Using STILE. *Proceedings of the 21st Annual Hawaii Int. Conf. On System Sciences*, 1988, Vol.2, pages 639-647.

Tenenberg, J. (1986). Planning with abstraction. In *Proceedings of AAAI-86, Fifth National Conference on Artificial Intelligence*. Philadelphia, PA.

Thompson, J. D. (1967). *Organizations in Action: Social Science Bases of Administrative Theory*. New York: McGraw-Hill.

Thomsen, K. S. (1987) Inheritance on processes, exemplified on distributed termination detection. *International Journal of Parallel Programming*, *16* (1), 17-53.

Ulrich, K. T. & Eppinger, S. D. (1995) *Product design and development*. McGraw-Hill: New York.

Ulrich, D. O. and McKelvey, B. (1990) General Organizational Classification: an empirical test using the United States and Japanese electronics industries, Organization Science, 1: 99-118.

Wegner, P. (1987). Dimensions of object-based language design. In *Proceedings of the Conference on Object-Oriented Systems, Languages, and Applications (OOPSLA '87)* (pp. 168-182). Orlando, Fla.: ACM.

Winograd, T. and Flores, F. (1986). *Understanding computers and cognition: A new foundation for design*. Norwood, NJ: Ablex.

Woodward, J. (1965). *Industrial organizations: Theory and practice*. London, New York: Oxford University Press.

Wyner, G., & Lee, J. (1995). Applying Specialization to Process Models. In *Proceedings of the Conference on Organizational Computing Systems*. Milpitas, California: Association for Computing Machinery, August.

Yourdon, E. (1989). *Modern Structured Analysis*. Englewood Cliffs, NJ: Yourdon.

Yu, E. S. K. (1992). Modelling organizations for information systems requirements engineering. *Proceedings IEEE*.

Zlotkin, G. (1995). "Coordinating Resource Based Depedencies." MIT Center for Coordination Science Unpublished working paper, March 1995.