KRESGE
BUS. ADM.
LIBRARY

RESEARCH SUPPORT
UNIVERSITY OF MICHIGAN BUSINESS SCHOOL

JULY 1996

DEPICTING THE USE AND PURPOSE OF DOCUMENTS TO IMPROVE INFORMATION RETRIEVAL

WORKING PAPER #9603-08

MICHAEL D. GORDON AND SCOTT A. MOORE
UNIVERSITY OF MICHIGAN BUSINESS SCHOOL

AUG 1 2 1996 ✓

Depicting the use and purpose of documents to improve information retrieval

Michael D. Gordon
Scott A. Moore
Computers & Information Systems Department
University of Michigan Business School
{mdgordon,samoore}@umich.edu

July 9, 1996

Abstract

In this paper we discuss a new kind of information system that helps people be ready for information work and locate documents. This system differs from a traditional information retrieval system by relying extensively on descriptions of both how a document is used and the purposes it is used for. A formal language represents this information.

1 Introduction

The discipline of information retrieval is focused on locating needed information. After several decades of research, the field has made many advances in both commercial and laboratory settings. Predominantly, these efforts have centered on describing the content of documents, using keywords or other representational systems; to a lesser degree, other efforts have concentrated on a document's contextual information, such as its author or publisher.

We claim that document descriptions can be strengthened and retrieval improved by paying attention to how documents are *used* and what their *purposes* are. We describe a formal language for describing and capturing such information. By "formal" we mean that the language has a defined syntax and grammar that allows it to be interpreted by a computer.

In §2, we describe the *readying problem*—the problem of making sure that information workers can effectively locate and make use of the information they require. In §3 and §4, we describe a formal language for describing work and in §5 we show how this language assists in addressing the readying problem and in improving retrieval. We compare this work with other research in §6 and summarize and discuss our findings in §7.

2 Readying

For an information worker, being ready to work takes many forms. For one, before she begins work, an information worker needs relevant information for the problem or assignment at hand. For another, while switching among tasks she must be able to resume work on those that are suspended or interrupted as effectively and effortlessly as possible. Finally, even after she completes a task, it may be important later for someone (even the original worker) to understand its conclusions, assumptions, sources of evidence, etc. So, being ready to work means locating and making sense of the information necessary for information work—before, during, and after it is conducted. We call this the *readying problem.* A system that provides this capability is a *readying system.*

In other words, the readying problem is concerned with how an information worker can eliminate or reduce the overhead and wasteful, non-value adding tasks that surround attempts to find, store, share, and make appropriate use of information. One way to reduce such overhead is through business process reengineering, an activity which often results in task reassignment and work redesign (Hammer & Champy [10], Harrington [12]). But even after reengineering a worker is never assured of locating needed information. Nor is she assured of uninterrupted work, or of being able to make sense of work already completed.

We contend that two activities can go a long way toward improving the readying problem: recording and documenting.

recording Recording means describing the events associated with a document's use. This means detailing when a document was created, modified, transmitted, and looked at, and who performed these events. Normally, such a history is never officially written down, though a chronology of these events can be useful in identifying needed information (Lamming & Newman [14]). In this paper we show how recording can *automatically* be performed and can be exploited for the purpose of more effectively locating documents.

documenting Documenting means describing the associations one has to a document. For instance, a document's author may annotate it either while writing it or after completing it. Documenting allows people who might later need the document both to locate it more effectively and to recall more easily its origin, use, limitations, context, etc. A worker may anticipate a specific need for a particular document: "I'll need to have this document at the next Winter Retreat so that I can explain things to the Board." Or the need may be more serendipitous: "This might be nice to hold on to; I'm just not sure when I'll need it again." The more carefully and completely the worker's mental state in relation to a document can be captured and documented, the more effectively she or others will be able to use that document in the future. Even the author of a document who resumes work on it after a considerable interruption may find she is unable to do so effectively unless she can recapture her thought processes while she was last working on it. Documenting can be as simple as using a designated folder for a project or can involve more elaborate means of making a document's context evident. Recording can even be considered a form of documenting if the resultant records remind a worker of people or events associated with a document.

Locating documents and making effective use of them (in terms of recognizing their impact, context, limitations, etc.) are the reasons for recording and documenting. Locating and making effective use of information can support activities deemed vital to the firm ("Find me all the information relevant to the decision to build the new manufacturing plant") or far more pedestrian. Generally, what is common to most attempts to locate information is that much needed information is out of file, not locateable, incomplete, or somehow different than what one had imagined; finding the information one truly requires can occupy up to 25% of one's day (Gordon [9]). Still, even though being able to easily locate needed information can greatly reduce the wasteful, non-value adding efforts that beset information workers, most rarely make a concerted effort to document or record. They consider these tasks peripheral to their main responsibilities. But, just as a building contractor attends to the "small task" of the timely delivery of mortar—to keep from delaying the mason and thus the plumbers and electricians, finally causing a project to miss its deadline—information workers benefit when attention is paid to the small tasks of recording and documenting by recapturing some of their lost ten hours per week.

In the next section, we describe the bases of a computer-processable lan-

guage that is used to record and document information work. The focus of the language is on the small tasks common to all information workers whether clerk, loan officer, bank vice president, or professor—creating documents, editing them, transmitting them, etc. Because these small activities can be described independently of what each profession uses documents for, a common language can be used to record and document work. Since this information can be automatically captured, simply doing work provides a basis for improved retrieval. We provide examples of this significant result in §5.

3 Foundations of a readying system

A readying system is system-level software that interacts with application software and other system software. Much of the time a readying system works unobtrusively, recording the actions of users as they use word processors, email systems, fax machines, printers, and other office equipment and applications. Work conducted manually and information that cannot be stored on a computer lie outside of our interest in readying systems.

A readying system can also perform various information tasks, including: creating, sending, or storing documents; responding to requests for information; and reminding one of deadlines or appointments. It does some of these activities automatically (for example, responding to a request for information similarly to the way an EDI system might automatically respond to a request for a price quotation). It partially automates others, relieving an information worker of certain responsibilities better suited to a computer, but not totally absolving her. In this sense a readying system is similar to an office automation (e.g., Malone et al. [15], Mora et al. [16], Sohlenkamp & Chwelos [21]) or a work flow system (e.g., Radosevich [19], Trammell [22], Wilson [25]).

In either case, as well as when an individual is simply using a computer without receiving any apparent support from her readying system, a readying system records and documents the work she is doing. In this paper, our primary aim is to show how a readying system offers a new form of information retrieval based on its ability to record and document, and we pay less attention to the way the system provides some level of automation.

Next, we describe the theoretical concepts that are the basis for a readying system.

Table 1 Information acts classified by type—Top two levels

Types	Acts
Use acts	create, open, begin-change, commit-change, close, destroy, retrieve
Annotative acts	link, make-note
Documentary acts	send, receive

3.1 Information acts

A readying system is based on a computer-processable, or formal, language for information work. The language's fundamental construct is an *information act*—an action that can be performed on or with a document or some other information object. Information acts can describe a document's use, its purpose, and how it has been annotated. Table 1 lists the first two levels of an information act hierarchy. Except for more detailed actions that cannot be shown in just two levels, it describes the full set of actions that can be performed on documents. Other typologies might do the same even though they include different acts, different numbers of levels, or different specialized acts. As we will see, information acts form the basis of our system's recording mechanism.

The first type of information act shown in Table 1, *use acts*, describes various acts involved in composing and examining a document. The create and destroy acts denote the creation and deletion of a document. Open and close denote opening and closing a document for the purpose of either modifying or reading it. Begin-change denotes that the document is no longer merely open—some change has been made to it. When these changes are committed to long term storage, a commit-change act occurs. Finally, retrieve denotes retrieving a document from an information base. Each information act in Table 1 will have more specialized cases, possibly several levels deeper than shown. For instance, create has create-email-doc and create-wordprocessor-doc as two of its specializations at the next level. A readying system can detect and capture each of these information acts (including annotative and documentary acts which we discuss next) in a database as an information worker uses her computer for e-mail, word processing, looking for files, etc. For instance, a *Save* command in a word processor would be represented by a commit-change.

Sequences of use acts more fully describe how a document is composed

and examined. For instance, a completed first draft of a document written in two editing sessions might be recorded as follows (with details deferred until §4): create-wordprocessor-doc, open, begin-change, commit-change, begin-change, commit-change, close, open, begin-change, commit-change, close, open, close.

Annotative acts are a second type of information act. They describe the acts involved in documenting work (i.e., keeping track of mental associations to documents). Two specializations of this type of information act are the following: link, which creates a pointer that suggests that one document refers to another; and make-note, which is a note on a document written to ensure that the document will be better understood and more useful.

A third type of information act is a *documentary act.* A documentary act is a communication act involved in sending or receiving a document. Documentary acts are analogous to speech acts. Speech act theory, a linguistic theory originally developed by Austin [2], states that when people say something (with words), they are actually *doing* something as well. Examples of this phenomenon are the sentences "I now pronounce you husband and wife," and "I find the defendant not guilty." In both cases saying so *makes* it so. Documentary acts are an analogous concept in our theory, where our focus is on what people can *do* with documents. So, while Austin used the phrase *speech act* to mean "doing something with words," we coin the phrase *documentary act* to mean "doing something with documents." In speech act theory, speech acts take effect only when a speaker says something, and a listener hears it. Analogously in our theory, we suggest that documents "do something" only when they are sent to one (or many) "listeners." *Broadcasting* refers to sending a message to two or more recipients. The receive documentary act is the act that occurs when the recipient receives the send documentary act.

Austin argued that there are many different kinds of speech acts, which differ based on their *illocutionary force.* This force defines the utterance's function or purpose—that is, what the utterance is supposed to accomplish. A speech act has the structure F(P), where F is its illocutionary force and P is the proposition it is applied to. For example, the utterance "Is it raining?" is a speech act with an illocutionary force of *question* and a proposition that *it is raining.*

Austin proposed a typology of illocutionary forces. Speech act theorists propose that, no matter what a person says, the statement can be meaningfully classified by a speech act typology, thereby equating innumerable sentences according to their illocutionary force—that is, according to their function. For present purposes we wish to describe a *document's* function

Table 2 Illocutionary forces used in our system

inform	retract	predict	confirm	question
require	permit	promise	offer	

when it is sent. When a user (or the readying system) performs a documentary act, she (or it) must pick the appropriate force, F, and supply the proposition, P. We discuss this process later.

The typology of illocutionary forces that we use is shown in Table 2. It is a subset of those proposed by Bach & Harnish [3] that Moore [18] found sufficient to handle many business transactions. Others (e.g., Ballmer et al. [4] or Searle et al. [20]) have devised their own typologies. We can add forces if the need arises to make our typology more expressive though Moore's work [18] suggests this should not be necessary.

Collectively, the speech acts in Table 2 express many of the things one can do by sending a document. For instance, a document may *inform* its recipient about some new development. It may *retract* (or *confirm*) ideas previously presented. It may *predict* some future state of affairs or present a *question* that needs addressing. Documents can also have more official functions: they may *require* that something occur, *promise* that something will occur, *permit* something to occur, or make an *offer*.

As with information acts, speech acts can be specialized in many different ways. For instance, "to remind" is a specialization of the speech act "to inform" that depends on the hearer already knowing what she is being told. "To allocate" is a specialization of "to promise" where the promise concerns a scarce resource.

To summarize what we have said in this section: We have introduced the idea of information acts—which we view as the smallest meaningful unit of description of how documents are used. Information acts describe 1) the ordinary things one does with a document (use acts); 2) the activities that take place as a worker attempts to make permanent certain associations she has to a document (annotative acts); and 3) the purposes documents are used for when they are transmitted to others (documentary acts). Information act descriptions of documents can supplement keywords or other types of descriptions that attempt to explain what a document is *about*.

A readying system unobtrusively records a worker's actions as if it were a camera with a special lens that only detects information acts. The readying system "camera" will record each type of information act. Jumping ahead,

some quick examples show how information acts can be useful in information retrieval. Responding to the query "It's the document I received from Sue just before I received the new strategic direction from Joe" depends on recording *what* one is doing with a document and when (use acts). Responding to the query "It's the document in which I requested that Sue be transferred" depends on keeping track of documentary acts, which record via speech acts a document's purpose (here, a request). And the query "It's a document that I annotated as contradicting the 3rd Quarter salary data" depends on annotative acts. We give fuller attention to such queries in §5.

3.2 Work processes

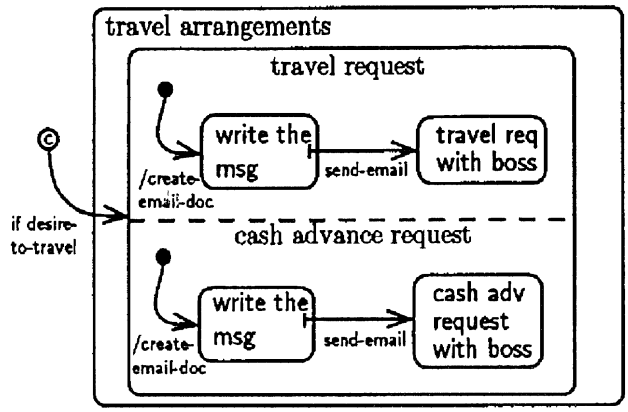
Meaningful work involves many individual information acts, each of which contributes to the same end. We call such sequences, or such compositions, of information acts *work processes.* Work processes are "chunks" of work that, in many situations, make more sense or, certainly, provide a more complete view of work than do isolated information acts. Work processes have two related roles in a readying system, 1) as a "program," which describes how to do work, and 2) as a record of what work has been done.

We use statecharts (Harel [11]) as the external representation of work processes that is visible to users of a readying system. (Internally, we represent the same information by a set of equivalent Prolog statements.) Statecharts are comprehensible to people and allow them to construct or to examine work processes. In the following examples, we show how statecharts serve both roles.

Figure 1 is an example of a statechart that can be used as a program. This work process is called travel arrangements and helps a worker get the paperwork done for a business trip. This program is invoked when a person expresses a desire to travel (say, by making the appropriate selection from a menu), thus causing the desire-to-travel condition to become true. Every program has a similar entry condition on the outside that determines when the program should be executed. The travel arrangements work process consists of two separate processes that are executed concurrently: the travel request process and the cash advance request process. A statechart indicates concurrency with a dotted line dividing a rectangle.

As its name suggests, a statechart depicts a sequence of states (indicated by rounded rectangles). The arrows entering a state indicate when a transition should be made into that state from another. An arrow that begins with a black circle shows the default starting point for a process. In this case, since there are two concurrent processes, there are two starting states,

Figure 1 A statechart representation of a work process



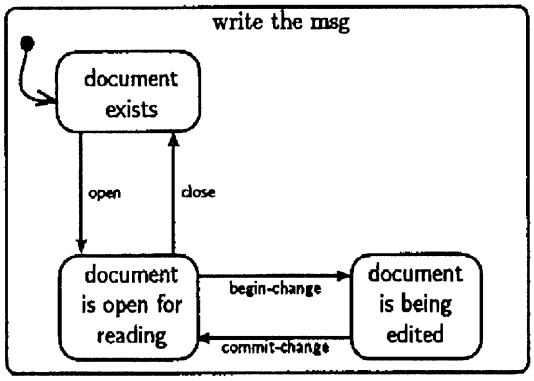
both called write the msg, that are entered at the same time.

Consider the arrow at the beginning of the travel request process—it has /create-email-doc below it which indicates that the system should execute the action called create-email-doc. Notice that create-email-doc, like other actions labeling transition arcs, is a specific instance of an information act from Table 1—in this case, the *create* act (one of the use acts).

The complete format of a transition label from state A to state Z is α if γ / β, which is interpreted as follows: "The process will go from state A to state Z when α occurs and if γ is true. If this transition takes place, then, simultaneously, the program will execute β." Each part of this label is optional. (As a notational convention, the if is used only when there is a γ part of a label and the "/" is used only when there is a β-part of a label.) In the case of the label /create-email-doc, the system automatically takes the transition since there is no condition on it and executes the create-email-doc act.

Information acts make up the α and β parts of the transition label while predicates that reflect a condition like the number of documents in an individual's electronic in-box or some state of the environment make up the γ-part. A transition with an information act in its β-part (e.g., /create-email-doc) represents an information act that a readying system automatically performs as a side effect when the α if γ condition on that transition is satisfied. The α if γ part of the transition acts as a sentinel, keeping the system from changing state. Once the act α is performed by either the worker (because she has the information, time, and wherewithal to perform

Figure 2 The "write the msg" work process



the act) or the readying system (because it was triggered by some other event), the process will move to a new state if γ is true.

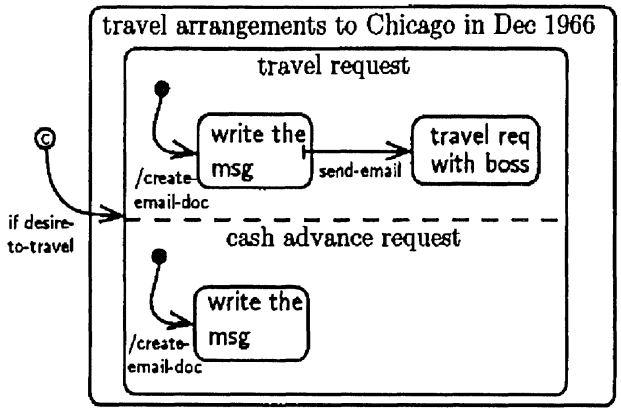
Returning to our example in Figure 1, in the write the msg state the worker is inserting text into the document. As the transition label from this state indicates, the system will leave this state when the user performs the send-email documentary act.

An important point about statecharts is revealed by the write the msg state (details shown in Figure 2); namely, writing the message may be done in one or more editing sessions, and it may be performed by typing, copying text from another document, or in other ways—we cannot tell from the statechart in this figure. Accordingly, there may be many individual information acts associated with the write the msg state that are not portrayed in the travel arrangements statechart. In fact, this virtue of statecharts—their ability to show details at various level of abstraction, even within a single diagram—motivated us to use them to represent work processes.

To complete our example, at the same time the system begins the travel request process, it also begins the cash advance request process. This process has the same structure as the travel request process but involves a different document that performs a different function.

Remember that there is a second role of work processes—as a record of work that has been performed. As work is completed, with or without some level of automated support from a readying system, the appropriate information acts are recorded. For example, Figure 3 shows a partially completed work process. As depicted, this figure looks very similar to a statechart that is used as a program. In actuality, it differs in two important

Figure 3 A statechart representation of a partially completed work process

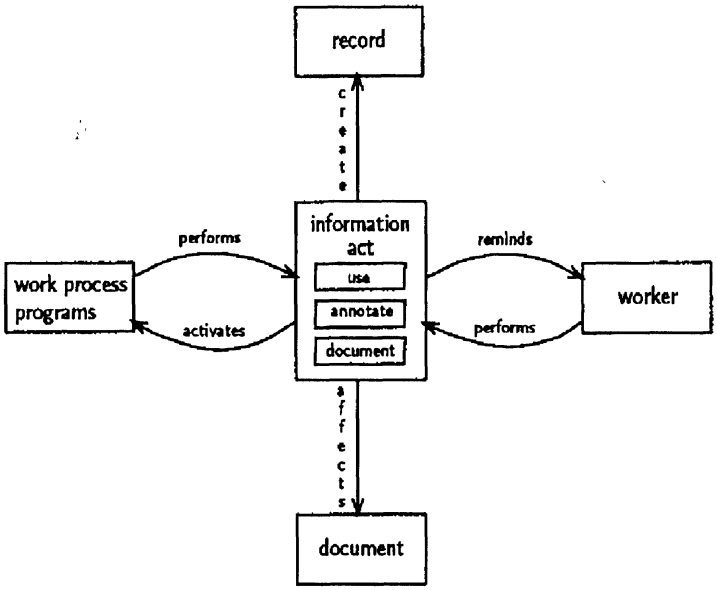


ways: First, it is incrementally composed, on-the-fly, as work is performed; in contrast, statecharts used as programs are composed in advance. For instance, in Figure 3 where travel arrangements are being made for a trip to Chicago, the travel request has been sent (since the current state in travel request is travel req with boss) but the cash advance message is still being composed (since the current state in cash advance request is write the msg). Second, a statechart used as a record contains much detail about the actions that were performed: who performed them, when they were performed, etc. These important details will be presented in section §4.

Not all work with documents fits into a predefined work process. But all information acts associated with all documents are still recorded by a readying system. Every time a worker creates, adds text to, edits, opens, closes, or sends a document, those information acts are recorded, even when they take place outside a programmed work process. In such situations a state chart representation links together information acts chronologically.

Figure 4 summarizes the interrelationship between people and work processes. While doing work that involves documents, a worker will perform information acts. These actions might either 1) activate some work process program which can, in turn, perform some other information act or 2) remind a worker to perform another information act. The readying system will create a record whenever an information act is performed. A worker may document (via annotative acts) any documents she has worked with. Subsequently, what is recorded and documented can improve a worker's ability to locate needed information.

Figure 4 Relations among uses of information acts



3.3 Information objects

Implicitly, we have been referring to documents that contain text throughout this paper. But there is nothing special about text in what we are describing. To make this point clear, what we have said so far—as well as all that follows—pertains to any *information object.* By this term we usually mean ordinary documents such as memos, meeting minutes, reports, and letters; but we can also mean drawings, scanned images, or anything else stored on a computer. That is, information objects generalize the concept of text-only documents, and are "documents" in the broadest sense of the term.

An information object has content and a context. The content of a document is the subject material it specifically expresses. For drawings, images, and other non-text information objects, the content is analogously defined. Context includes information *describing* the document that is not part of its content proper. For instance, the content of meeting minutes is what was written in that document. Its contextual description includes the document's author, the meeting time, the meeting location, the attendees, the leader of the meeting, the financial position of the company at that time, the news of the day, etc. The context also includes information about work:

what project is the meeting part of, who called the meeting, what event precipitated the meeting, what resulted from the meeting, etc.

Content descriptions (exemplified by keywords) are based on human perceptions (which can vary widely among users) and are subject to linguistic problems surrounding synonyms, homonyms, polysemy, etc. On the other hand, contextual descriptions are based on facts, and so can be advantageous for identifying and retrieving documents or other information objects. As we will discuss in the following section, we have defined a detailed description of how information acts should be recorded so that some of this contextual information is automatically captured whenever information work is done. It is straightforward to modify or extend what can be recorded to include other contextual information.

4 Recording work with information acts

A simile we used before to describe a readying system was that it records office work as would a camera with a special lens that only perceives information acts. We have mentioned that much detailed information is recorded that we did not portray in our statechart representation. Here we explain these details.

As a point of emphasis, remember that the transitions in any statechart are of the same structure: each contains an information act or other observable conditions. The *states* within a statechart will vary dramatically, of course, from work process to work process. So, a technical, legal work process, for example, will not have a travel req with boss state as would as would a travel work process, but will have states appropriate to some aspect of technical, legal work. Still, all work processes are built from the same information act building blocks. In this way, we have a descriptive language capable of describing any kind of document used in any kind of information work. It is precisely this fact that we exploit for retrieval.

4.1 A grammar for work

A readying system records much information about an information act by adhering to a context-free grammar expressed in BNF. By recording information acts with this grammar, it is possible for a retrieval system to parse them (and their compositions as statecharts) and so use them to locate documents more effectively. The grammar for information acts is shown in Figure 5.

Figure 5 Partial BNF representation for recorded information acts

```
1. infoAct ::== "infoAct(" actor "," theAct "," iaContext "," iaID ")"
2. theAct ::== useAct | annotativeAct | documentaryAct
3. useAct ::== createAct | ("open(" docID "," application-name ")") |
   beginChange | ("commit-change(" docID ")") ("close(" docID ")") |
   ("destroy(" docID ")") | ("retrieve(" docID ")")
4. annotativeAct ::== ("link(" docID "," docID ")") |
   ("make-note(" docID "," plainText ")")
5. documentaryAct ::== send | receive
6. createAct ::== ("create-email-doc(" docID ")") |
   ("create-wordprocessor-doc(" docID ")")
7. beginChange ::== ("begin-typing(" docID ")") |
   ("copy-change(" from "," to ")")
8. send ::== ("send-email(" recipient "," speechAct "," docID ")") |
   ("send-fax(" recipient "," speechAct "," docID ")")
9. receive ::== ("receive-fax(" sender "," speechAct "," docID ")") |
   ("receive-email(" personID "," speechAct "," docID ")")
10. speechAct ::== ("speech-act([" 1#("sa(" force "," content ")") "])") |
   ("speech-act([" 1#("sa(" force "," contentDesc ")") "]," docID ")") |
   ("speech-act([" 1#(force "(null)") "]," docID ")")
11. iaContext ::== "[" #( ("process-id(" processID ")") |
   ("time-of-act(" timePred ")") | ("respond-to(" iaID ")") |
   ("step(" stepID ")") | ("project(" projectID ")") ) "]"
12. force ::== "inform" | "retract" | "predict" | "confirm" | "question" |
   "require" | "permit" | "promise" | "offer"
13. actor ::== personID | ("actor(" (personID | "system") "," personID ")")
```

Missing terms (e.g., timePred) are defined in a straight-forward manner. We have limited our presentation to those terms necessary to understand the basic structure of the language.

The augmented BNF used in this definition is defined in RFC 822. Literals are surrounded by quotes while terms are not. The bar (|) means "or" while the square brackets indicate optionality. Elements enclosed in parentheses should be treated as a single element. A pound sign (#) is used to indicate a list of terms; thus, 1#(person) indicates a list of 1 or more persons.

In the first rule of the BNF, an information act (infoAct) is defined as an actor performing a certain information act (theAct) within a context (iaContext). The last symbol in the definition (iaID) is the identifier for the information act. As shown in the second rule, theAct is allowed to take on one of three possible values: use acts, annotative acts, and documentary acts. These are the three types of information acts previously shown in Table 1. The next three rules define the specifics for each type of information act.

As explained in §3.1, documentary acts include a speech act (rules 5, 8, 9). Recall that a speech act's structure is F(P), where P is the *argument* of the illocutionary force F (see rule 10). The list of forces associated with a speech act (see rule 12) are those from Table 2.

The illocutionary force, P, used with a speech act (rule 10) defines speech acts of two main types. First, the speech act can be a fully computer-processable message that does not refer to any external document; a simple example is

```
(1) infoAct(molly,
        send-email(lindsey,
            speech-act([sa(inform,
                snow(city(ann-arbor, mi, usa),
                    time("03:30:00p on May-12-1966"))])],
            none),
        [time-of-act("03:33:13p on May-12-1966")],
        i21)
```

This is an email message from molly to lindsey in which molly is informing lindsey that it is snowing in Ann Arbor at 3:30pm on May 12. Such formal messages, which use predefined predicates, are especially useful for allowing work process programs to parse and automatically respond to incoming messages.

The other type of speech act refers to accompanying documents. Examples of the three forms of this type follow. If a person were to send a document about the weather in Ann Arbor to another person, this speech act would look something like

```
(2) infoAct(molly,
        send(lindsey,
            speech-act([sa(inform, weather)]),
            d32),
        [time-of-act("03:33:14p on May-12-1966")],
        i22)
```

This is a simpler message than in (1), using free-text for the argument of inform, but it is less informative. It tells the recipient that the purpose of document d32 is to inform about the weather. The message sender could

also substitute sa(inform, "It is snowing") for sa(inform, weather). Finally, the third form of this type of speech act is even less informative; it merely tells the recipient what illocutionary force(s) the document carries. An example of this type of message is

```
(3) infoAct(molly,
        send-email(lindsey,
            speech-act([inform(null)]),
            d32),
        [time-of-act("03:33:15p on May-12-1966")],
        i23)
```

Though these types of speech acts vary in their specificity, each can be useful for retrieval as we will see in §5.4.

Different readying systems might be based on a slightly different grammar. For instance, in Figure 5 the iaContext term only includes the work process associated with the information act, the timing of the act, the information act that this information act is a response to, the particular step of a work process the act is associated with, and the project associated with the information act. We have chosen these items on the assumption that they will often facilitate retrieval. Similarly, the speech act typology we have used in Table 1 (which makes up the force term in Figure 5) could be replaced by another typology; and other information acts could be produced by different production rules. For different types of work environments, these changes might make a readying system more useful and retrieval more effective.

5 Improving information retrieval

In this section, we show how recording information acts can improve information retrieval. In doing so, we will also illustrate some of the details about recorded information acts, as well as how work processes used as programs provide some level of work automation.

5.1 Retrieval based on facts

As we just saw, for any information act some contextual information is recorded, including the actor (the person performing the act) and the time of the act. Recording these contextual facts in a formal way in a database allows certain elementary, fact-based questions to be answered by a query against the database.

For instance, to answer the question "Which documents did Phil create?", a searcher would issue the formal query

```
(4) Find DocID where
    infoAct(actor = phil,
        theAct = create(DocID))
```

(The clauses within the infoAct term restrict the information acts that might ultimately match the query.) The retrieval system would then consult its database of information acts to find matching information acts, such as

```
(5) infoAct(phil,
        create(d72),
        [time-of-act("11:42:16a on Oct-10-1966")],
        i35).
```

This information act can be interpreted as "Phil created document d72 at 11:42:16a on October 10, 1966. The identifier for this act is i35." The complete answer to the query would be a set of document identifiers (**DocID**) gathered from information acts in which Phil creates a document; this set would include document d72. Questions like "Which documents did Phil create in August, 1996?" and "Which documents were created in August, 1996?" could be just as easily answered.[1]

Ordinary file systems can answer some of these questions, too, though not necessarily easily. However, since we record with information acts all actions on documents, we can directly answer questions that few other systems could. For instance, "Which documents have been written or revised on the system by two or more people?" is answered by finding any documents that have at least two commit-change information acts performed by different people:

```
(6) Find Doc where
    infoAct(theAct = commit-change(Doc),
        actor = P1),
    infoAct(theAct = commit-change(Doc),
        actor = P2),
    P1 ≠ P2
```

Even more difficult (and likely impossible) for most other systems to respond to would be a question like "Who has seen the document that Dennis wrote on January 1, 1981?" From the database of recorded information acts, we simply find the actors in the open information acts (as specified below) that satisfy the following compound query:

[1]For the rest of this paper, we will continue to define a formal query using an SQL-like query language (as in (4)) that highlights salient attributes. However, in showing an information act, we will substitute ellipses (...) for its non-essential parts to make our examples easier to understand. Further, some acts may not be listed in Table 1 because they are specializations of those acts.

```
(7) Find P where
    infoAct(actor = dennis,
        theAct = create-wordprocessor-doc(Doc),
        date-of-act = "Jan-01-1981"),
    infoAct(actor = P,
        theAct = open(document = Doc))
```

This query finds the people **P** who have performed the information act **open** on any document **Doc** that was created by Dennis on January 1, 1981. Of course, if Dennis created more than one document on that date, we would need to restrict the **create** information act further if we wished to find out about a specific document. Questions similar to these can be especially useful in identifying a document known to have been seen by certain individuals.

Information acts may also directly record that one document is related to another. For instance, suppose that Mackenzie reads document **d25** from Hannah and thinks that it is related to the Oracle document **d6**. Mackenzie can manually link these two documents, thus producing the information act:

```
(8) infoAct(mackenzie
        link(d6, d25),..., i41)
```

To find documents that have been manually linked to document **d6** we submit the following query:

```
(9) Find Doc where
    infoAct(theAct = link(d6, Doc))
```

"Chains" of linked documents can be useful for uncovering associated information that is not contained within a single document. Such chains can be found by retrieving a given document plus all others that are transitively linked to or from it by either **link** or **make notes** information acts. Similar capability is generally unavailable in traditional file systems.

The system is also capable of inferring links when they are implicit, such as when a message is composed in response to another message. For example, suppose that Mackenzie's system records the following information act signifying that it received a question via e-mail from Hannah:

```
(10) infoAct(mackenzie,
         receive-email(hannah,
             speech-act([question(null)]),
             d25),
         [project(kennedy),
             process(define-project),
             step(set-requirements),
             time-of-act("03:06:58 on Oct-26-1966")]),
         i36)
```

Mackenzie's system could automatically create a document that would be completed by Mackenzie. The system could ensure that any contextual information associated with the act in (10)—and, hence, with document d25 referred to within that act—is automatically associated with the document created in response to that act. At a basic level, this means that the act that automatically creates the document contains a respond-to predicate:

```
(11) infoAct(actor(system, mackenzie),
         create-email(d73),
         [respond-to(d25), ...],
         i37)
```

This effectively establishes a link between d25 and d73. But, more interestingly, any information about the original email (such as its project) can be incorporated into the automatically composed response. Thus, the previous information act might actually look like:

```
(12) infoAct(actor(system, mackenzie),
         create-email(d73),
         [respond-to(d25),
             time-of-act("03:07:15p on Oct-26-1966"),
             project(kennedy), process(define-project),
             step(set-requirements)],
         i37)
```

The advantage here, of course, is that we can easily create *ad hoc* conceptual categories, like the "Kennedy project" or the "define-project process," and then find "linked" documents that are elements of such categories.

Another form of retrieval based on facts comes from having knowledge about software packages. For example, the question "Where is the word processing document that contains information copied from a graphics package?" might be asked as follows:

```
(13) Find Doc where
     infoAct(theAct = create-wordprocessor-doc(Doc)),
     infoAct(theAct = create-graphics-doc(Doc2)),
     infoAct(theAct = copy-change(Doc2, Doc))
```

Annotation acts also allow retrieval based on facts. By recording annotations, such as

```
(14) infoAct(hank,
         make-note(d22, "Need Don's input before continuing."),
         [time-of-act("03:01:22 on Oct-26-1966")],
         i35)
```

we can answer questions like "Where is the document Joe wrote but that Hank annotated?":

```
(15) Find Doc where
     infoAct(actor = joe,
         theAct = create-wordprocessor-doc(document = Doc)),
     infoAct(actor = hank,
         theAct = make-note(document = Doc))
```

Similarly, whenever one is working with a document and hopes to remember some of its broad, underlying context, one can ask to see all the annotations that have been associated with it.

5.2 Retrieval based on timing relationships

Dates and times are sprinkled throughout our examples. Sometimes, by means of deductive inference, timing relationships can be the key to describing the information one wants to retrieve. For instance, consider the need for information expressed by

> "Find the document Maria wrote about the Kennedy project. Kathryn received it just before the 'Right sizing' document was broadcast by Joe to all employees"

A query that would find this would be:

```
(16) Find Doc where
     infoAct(actor = maria,
         theAct = create-wordprocessor-doc(Doc)),
     infoAct(actor = kathryn,
         theAct = receive(Doc),
         date-of-act = Date1,
         project(kennedy) in iaContext),
     infoAct(actor = joe,
         theAct = send-email(recipient = all-employees,
             document = Doc2),
         date-of-act = Date2),
     Date2 - Date1 ≤ 1-day,
     "right sizing" in subject(Doc2)
```

The success of our method hinges on the fact that our system records the dates and actors of the relevant information acts and can link these with the facts about documents (as exemplified by the last two lines in (16)).

Another form of question one might raise—and one which also relies on deductive inference other systems are unable to support—is "I (Leigh) am looking for the document I was writing about the Nooton project that I revised again and again until I liked it." A suitable query would examine each document that was part of the Nooton project. It would count the number

of times each document was revised, and present them to the inquirer sorted on that basis:

```
(17) Find Doc, Count where
     infoAct(actor = leigh,
         theAct = create-wordprocessor-doc(Doc),
         project(nooton) in iaContext),
     Count = count(infoAct(actor = leigh,
         theAct = commit-change(Doc)))
     sorted by Count
```

5.3 Retrieval based on a hierarchy of information acts

In §3.1 we mentioned that information acts are organized hierarchically. This provides an opportunity for flexible retrieval, because we can ask for documents in a specific way and still retrieve them even if certain details are inaccurate. For instance, "Find the faxes that David received from Anne" would be specified by the query:

```
(18) Find Doc where
     infoAct(actor = david,
         theAct = receive-fax(actor = anne, document = Doc))
```

(This assumes that faxes are retrieved over computer fax-modems and therefore recorded by information acts.) If the system determines that David received no faxes from Anne, it can reason that receiving by fax and receiving by email are both specializations of the receive information act, and so look for documents David received from Anne by email instead of fax. The details about how a particular querying system performs partial matching by generalizing or specializing information acts are not important here.

5.4 Retrieval based on speech acts

Recall that illocutionary forces are associated with any documentary act (an information act in which one tries to do something with a document by sending it). By recording these "purposes" for transmitting documents, new kinds of retrieval are possible. For instance, the question "Where is the request that Hannah sent to Mackenzie in October?" specifically relies on the fact that the document being sought was making a request (rather than having another illocutionary force). Documents with the right sender, recipient, and illocutionary force will be retrieved.

Multi-document exchanges of information can also be identified, based on illocutionary forces. For instance, consider

"Where is the document Hannah faxed Mackenzie? It was a confirmation of a question Mackenzie emailed her about being able to hire a new contractor. Find the original question, too."

The document sought (a fax) will have a force of confirm and a respond-to condition that associates it with a question Mackenzine raised by email. The relevant query would be the following:

```
(19) Find Doc1, Doc2 where
     infoAct(actor = mackenzie,
         theAct = send-email(actor = hannah,
             question in speechAct,
             document = Doc1)),
     infoAct(actor = hannah,
         theAct = send-fax(actor = mackenzie,
             confirm in speechAct,
             document = Doc2),
         respond-to(Doc1) in iaContext)
```

Illocutionary forces are arranged hierarchically. We suggest a base set of forces (Table 2) that have proven effective thus far (Moore [17]). But, each of these forces can be extended in many different ways so that they become the "parents" of more specialized forces. For instance, an illocutionary force of permit may have these specializations: *approve*, *regulate*, or *authorize*. To offer may be specialized by to *quote* (as in to quote a price) or by to *suggest*. An illocutionary force attached to a documentary act may be either a base force or a more specialized force. In general, users of a readying system must provide the appropriate force (since only they understand a document's purpose). The only exception is when the readying system is engaged in a completely formal conversation, like responding formulaically with an offer to a formal request for a price quotation.

As with information acts, the hierarchical organization of illocutionary forces provides benefits for retrieval. For instance, in the database of recorded information acts might be

```
(20) ...send-email(...sa(inform, "new vacation schedule"...)...
```

Inform is a base illocutionary force (by our categorization), and remind is a specialization of informing in the sense that one is informed about something again. So, a question "Find the reminder that was sent about the new vacation schedule" could uncover documents (like the one sent in (20)) informing (not reminding) about the vacation schedule. Again, the details about how a particular querying system performs partial matching by generalizing or specializing are not important here.

Finally, each of the various types of propositions we associate with illocutionary forces may provide some advantage for retrieval. Recalling an example we used earlier, each of the following is an acceptable example of a force applied to a proposition: inform(rain(city(ann-arbor, mi, usa), time("03:30:02p on May-12-66"))), inform("It is raining"), inform(weather), and inform(null). Whereas full propositions (like "it is raining") are most descriptive about the purpose of a document, a content fragment (weather), or a null proposition may be useful for retrieval, too. The query "Find documents informing Joe that it is raining" would find a document associated with the second speech act:

```
(21) Find Doc where
     infoAct(actor = joe,
         theAct = receive-email(document = Doc,
             inform("It is raining") in speechAct))
```

The query "Find documents informing Joe about the weather" would find the a document associated with the third speech act:

```
(22) Find Doc where
     infoAct(actor = joe,
         theAct = receive-email(document = Doc,
             inform(weather) in speechAct))
```

Neither query, however, would find both such documents, since we don't employ any semantic devices that relate rain to weather, for instance. The usefulness of a speech act characterization of document usage is that we can distinguish documents making, say, requests, about the weather from those that inform us about the weather. An inquirer (Diane) may even wish to view all the e-mail messages that make a request of her, which she would do as follows:

```
(23) Find Doc where
     infoAct(actor = diane,
         theAct = receive-email(document = Doc,
             request in speechAct))
```

5.5 Retrieval involving work processes and work

Documents can also be located according to their association with work process programs. For instance, to find out what documents are involved in making travel arrangements, one would inspect the travel arrangements statechart (shown in Figure 1). From this, it is immediately seen that the appropriate documents are two types of email messages: the travel request

per se, and the request for a cash advance. A readying system's knowledge about work process programs permits another type of query that compares expected and actual behavior. For example, the question "What work process is incomplete because Joe hasn't finished the paperwork?" would, implicitly, compare the record in Figure 3 (assumed to be describing Joe's work) with the work process program in Figure 1. The record (Figure 3) would indicate that the cash advance was created but never sent to the boss (as Figure 3 would require for the entire travel arrangements process to be completed).

6 Other research

There is a huge literature on office automation systems, which take a variety of approaches to supporting information work: a problem oriented approach [27], a procedure oriented approach [8], or even a knowledge based approach [23]. Similarly, there is a huge information retrieval literature. However, the intersection of these two bodies of literature, which are the bases for our work, is small.

Celentano et al. [6, 7] proposed a system for retrieving documents based on their uses and roles. A chief characteristic of this work is the need to specify in advance a document's use (who will prepare it, review it, etc.) and any special circumstances that pertain to it (for instance that a special letter of authorization is required only when a loan is at least $100,000). Retrieval of documents whose use or roles cannot be specified in advance lie outside the scope of this system.

Work done by Lamming and Newman [14] and Lamming and Flynn [13] also attempts to retrieve documents based on the activities they are involved in. These efforts attempt to provide complete, detailed activity logs for people, documents, office equipment, etc. For instance, the whereabouts of people wearing active badges that emit infrared signals are continually tracked throughout the day. Similarly, individual stylus-strokes on Personal Digital Assistants are recorded and time stamped. Later, these details are analyzed in an attempt to portray larger, more meaningful episodes of activity.

Our work thus falls somewhere between Lamming's and Celentano's: We use information acts as primitive actions on documents that, nonetheless, provide more meaningful descriptions of work than stylus-strokes or other very specific actions. On the other hand, information acts can be recorded and examined outside of predefined patterns of use (though they may also

be examined as part of work process programs when required).

Finally, speech act theory and other theories of communications have been proposed as ways to describe office transactions (Aüramaki et al. [1],Winograd and Flores [26]). Primarily, these works try to describe the commitments and obligations inherent in office work, rather than providing any application to information retrieval.

7 Summary & discussion

We have defined a new type of information system in this paper: a readying system, which makes an information worker ready to do work. We claim that information workers from all professions work with documents in similar ways and that a readying system can help them automate certain behaviors and also find the information they need to do their work. Our focus has been on the way a readying system provides a greatly enriched description of the way documents are used, thus providing an opportunity for new types of information retrieval queries. In essence, these descriptions of documents' use greatly expand the ordinary contextual descriptions of documents. Barfield [5] has complained that contextual information in the digital world is lacking and that this makes finding information more difficult than it should be. Our paper addresses this shortcoming.

To support querying in this enriched context, a readying system must *record* and *document* what has been done. We have devoted considerable attention in this paper to the bases for *recording*, including information acts, speech acts, and work processes. We have defined a formal language that should be considered a working prototype—it is complete and provides needed functionality but also can be modified given new demands by its users. For example, the open use act records which application opens a document. This information could be expanded: other acts could capture this information or applications could be grouped by functionality. Information about applications could also be removed if document-centric computing (suggested by technologies such as OpenDoc or OLE) becomes a reality. Other examples of potential changes to the language include providing document version control and allowing the user to declare that documents are complete and cannot or should not be revised. Thus, while the recording capability has many useful features, it can be extended as need be.

Documenting work involves extending the description of the use of documents even further. When one writes a document, there are countless unwritten associations she has to it: who is likely to approve of what it says;

what sources of data still need to be investigated; what arguments might be raised in objecting to its conclusions (and what counter-arguments might be raised to defeat them); and so forth. When a document's author returns to a document (after a lengthy delay) to continue working on it, these associations may be very hard to recapture, making it difficult to continue work. For the reader of the document, some of the author's implicit associations will not be evident, and the reader may misapprehend its relevance altogether.

A readying system is supposed to reduce the difficulties an information worker has in dealing with completely different information as she switches among many projects. Ideally, when one resumes a task, a readying system would effectively capture the mental "state" she was in when the task was suspended. An information act description of work, comprised of use acts and documentary acts, may help one "recapture state" in a limited way. Annotative acts help a bit more. But a truly effective readying system should support better an information worker's need to *document* her mental associations to a project (the arguments, counter-arguments, etc. we mentioned). Much research needs to be done to improve upon the customary devices people use today when attempting this, like attaching notes to documents or grouping documents together in a folder. Providing effective ways to document these underlying thoughts about a document's history and use may be extremely effective in helping information workers be ready to work.

Acknowledgments

File: getting-ready.tex. The authors contributed equally to this paper. Send correspondance to either author.

References

[1] AURÄMAKI, E., LEHTINEN, E., AND LYYTINEN, K. A speech-act-based office modeling approach. *ACM Transactions on Office Information Systems 6*, 2 (April 1988), 126–152.

[2] AUSTIN, J. L. *How To Do Things With Words*, 2nd ed. Harvard University Press, 1975.

[3] BACH, K., AND HARNISH, R. M. *Linguistic Communication and Speech Acts.* MIT Press, 1979.

[4] BALLMER, T., AND BRENNENSTUHL, W. *Speech act classification.* Springer-Verlag, 1981.

[5] BARFIELD, L. Sticky labels. *SIGCHI Bulletin 28*, 2 (April 1996), 95.

[6] CELENTANO, A., FUGINI, M., AND POZZI, S. Querying office systems about document roles. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval* (1991), A. Bookstein, Y. Chiaramella, G. Salton, and V. Raghavan, Eds., ACM SIGIR, pp. 183–189.

[7] CELENTANO, A., FUGINI, M. G., AND POZZI, S. Knowledge-based document retrieval in office environments: The Kabiria system. *ACM Transactions on Information Systems 13*, 3 (July 1995), 237–268.

[8] CROFT, W., AND LEFKOWITZ, L. Using a planner to support office work. In *Proceedings of Conference on Office Information Systems* (March 1988), R. B. Allen, Ed., ACM SIGOIS & IEEECS TC-OA, ACM Press, pp. 55–62.

[9] GORDON, M. D. It's 10 a.m. Do you know where your documents are? The nature and scope of information retrieval in business. *Information Processing and Management* (to appear).

[10] HAMMER, M., AND CHAMPY, J. *Reengineering the corporation.* HarperBusiness, 1993.

[11] HAREL, D. Statecharts: A visual formalism for complex systems. *Science of Computer Programming 8* (1987), 231–274.

[12] HARRINGTON, H. J. *Business process improvement.* McGraw-Hill, 1991.

[13] LAMMING, M., AND FLYNN, M. "Forget-me-not": Intimate computing support of human memory. In *Proceedings of FRIEND21: 1994 International symposium on Next Generation Human Interface* (1994).

[14] LAMMING, M. G., AND NEWMAN, W. Activity based information retrieval—technology in suport of human memory. In *Information Processing 92: Personal computers and intelligent systems* (1992), vol. 3, Elsevier, pp. 68–81.

[15] MALONE, T. W., LAI, K.-Y., AND FRY, C. Experiments with Oval: A radically tailorable tool for cooperative work. *ACM Transactions on Information Systems 13*, 2 (April 1995), 177–205.

[16] MEDINA-MORA, R., WINOGRAD, T., FLORES, R., AND FLORES, F. The action workflow approach to workflow management technology. In *Proceedings of the Conference on Computer-Supported Cooperative Work* (1992), J. Turner and R. Kraut, Eds., ACM SIGCHI & SIGOIS, ACM Press, pp. 281–288.

[17] MOORE, S. A. A communication framework for applications. In *Proceedings of the Hawaii International Conference on System Sciences* (January 1995), J. F. Nunamaker, Jr. and R. H. Sprague, Jr., Eds., vol. III, IEEE Computer Society Press, pp. 330–341.

[18] MOORE, S. A. Testing speech act theory and its applicability to EDI & other computer processable messages. In *Proceedings of the Hawaii International Conference on System Sciences* (1996), IEEE Computer Society Press.

[19] RADOSEVICH, L. Going with the flow. *ComputerWorld* (April 10, 1995), 87–97.

[20] SEARLE, J., AND VANDERVEKEN, D. *Foundations of Illocutionary Logic.* Cambridge University Press, 1985.

[21] SOHLENKAMP, M., AND CHWELOS, G. Integrating communication, cooperation, and awareness: The DIVA virtual office environment. In *Proceedings of the Conference on Computer Supported Cooperative Work* (October 22–26 1994), R. Furuta and C. Neuwirth, Eds., ACM SIGCHI & SIGOIS, ACM Press, pp. 331–343.

[22] TRAMMELL, K. Work flow without fear. *Byte* (April 1996), 55–60.

[23] TUENI, M., LI, J., AND FARES, P. AMS: A knowledge-based approach to tasks representation, organization and coordination. In *Proceedings of Conference on Office Information Systems* (March 1988), R. B. Allen, Ed., ACM SIGOIS & IEEECS TC-OA, ACM Press, pp. 78–87.

[24] WELLNER, P. Interacting with paper on the DigitalDesk. *Communications of the ACM 36*, 7 (July 1993), 87–96.

[25] WILSON, L. All together now. *InformationWeek* (November 28, 1994), 57–64.

[26] WINOGRAD, T., AND FLORES, C. F. *Understanding Computers and Cognition.* Ablex Publishing Corp., Norwood, NJ, 1986.

[27] WOO, C. C., AND LOCHOVSKY, F. H. Integrating procedure-automation and problem-solving approaches to supporting office work. In *Office Systems: Methods and Tools* (1987), G. Bracchi and D. Tsichritzis, Eds., IFIP TC8/WG 8.4, North-Holland, pp. 17–32.