Enterprise knowledge sharing, activity management, and a fabric for commitment

by R. Retallick
S. Sánchez

Line-of-business applications and contact and activity management applications have developed independently of each other. To integrate the two aspects of business communications, we need certain information structures with specific behavior that can be used by developers building either kind of application. The San Francisco™ frameworks provide us with the opportunities to develop these common business frameworks and make them available to other application developers. A set of object classes is described that provide a base for knowledge applications for use with San Francisco.

Enterprise computing has historically been justified by displacing people from business accounting functions. The resulting line-of-business (LOB) applications today are typically mature, integrated systems.

In the 1990s, development has been increasingly focused on communication, cooperation, and collaboration—the knowledge applications. Many of the data in this area are unstructured, and many different types of applications exist. Most of the applications have evolved from manual procedures, and their designs reflect their origins.

A more universal approach is needed to allow efficient data sharing and the sharing of tasks among many users in the enterprise. The new approach needs to support potentially high transaction rates and allow close interaction with traditional LOB functions.

By exploiting commonalities that exist in many of the data, a few simple data structures can bring order out of the chaos and support the various types of knowledge applications from a common set of objects. This paper describes a set of object classes for contacts, activities, notes, and topics that provide a foundation for knowledge applications within the IBM San Francisco* project.[1]

Historical perspective

Business data are of two types—structured data, such as accounting and product data, and unstructured or informal data, such as notes on human interactions, promises, and emotions. When business data were recorded with pen and paper, the informal data were recorded along with the structured data as margin notes and footnotes.

The advent of computers brought about the creation of LOB applications—mission-critical systems in which structured accounting data are stored centrally and managed carefully. These systems provide little

 0018-8670/98/$5.00 © 1998 IBM

or no support for the recording and retrieval of informal data. In the 1970s, the most common objection to replacing ledger cards with computers was the loss of the handwritten notes. Of course, the informal data, being in many ways just as important as the formal data, lived on, now separated from the structured data and managed inconsistently at the level of the individual user.

So it remains today. Although personal computing has improved many recording and retrieval processes for informal data, two characteristics persist:

1. It is dispersed and managed by individual users who may have come to regard themselves as the "owners" of the data.
2. A wide variety of separate PC applications is needed. For example, systems exist to manage calendars, faxes, e-mail, document creation, voice mail, tickler files, contacts, and projects, but a great many gaps and much overlap exist between the applications, leaving the user with the task of manually pulling them all together. As the user obtains better facilities, such as more powerful PCs, and more diverse applications, there have been few complaints. But this array of systems is not serving either the user or the enterprise well.

Requirements

Bob Buckman is CEO of Bulam Holdings, parent company of Buckman Laboratories, a worldwide chemical company. Bob hurt his back a few years ago and while lying in bed at home he pondered a recent valuation of his company that was millions more than the value of the assets.

Bob came to the profound realization that this value lay in the knowledge his employees had—thousands of dollars worth per person! He created a new *knowledge transfer department* and today spends as much or more on knowledge as on research and development, which for a chemical company is extraordinary. Bob says, "As we move towards the chaos of the future, the progress of Buckman Labs relative to other companies will be determined by the growth in the value of knowledge that exists in this company."[2]

Buckman Laboratories was the 1996 national winner of Arthur Andersen's "Best Practices" award in the category of "Knowledge Sharing."

Several kinds of knowledge exist in an enterprise. Knowledge about a product or process is the first kind of knowledge that springs to mind, and knowledge applications exist in the marketplace. These applications attempt to capture this kind of knowledge using inference engines to learn from expert users how they do their work. A less obvious form of knowledge is knowing what is going on in an organization. This knowledge does not just live in the mind of experts but, rather, portions of it reside in the minds of everyone who works in the enterprise. It is the lack of this knowledge that leads to mistakes, to frustrated customers who are tired of having to restate their problem every time they call a vendor and get a different person on the phone, to duplicated work, and to missed commitments. A third, even less obvious but no less important, kind of knowledge is something called "tacit knowledge." These are emotions, insights, intuition, and hunches. Ikujiro Nonaka in an article for *Harvard Business Review* speaks of using this tacit knowledge to create innovations.[3]

To share knowledge, we must do one of two things: store all relevant data centrally or distribute multiple copies. Distributing multiple copies only makes sense when there is no effective framework for storing the information centrally. The key word is "effective." It means efficient ways of capturing information already known to the users, and simple but powerful ways of retrieving the information. Such an arrangement means:

- *Marrying LOB and knowledge applications*—We need to repair the historical gap between LOB and knowledge applications. We need to move beyond the dark ages of interfacing and duplicating data into an integrated suite of applications. It is important to provide an organized interface. On occasion, the LOB application will need to trigger an activity. For example, an overdue payment could trigger an automatic collection call or a series of calls and faxes, all of which need to be recorded. On other occasions, a user may take a call and need direct access to the proper record in the LOB application for a shipping inquiry or order entry.

  The need to trigger such actions as sending collection letters after a delinquency date is recognized by LOB application developers, but they do not have available the entire support and communication structure of an activity management system, and so there are such cases where collection letters are sent to the customer who owes $19.95 and who has not only paid, but has just placed a million-dollar order. It is not enough to trigger an

event based on some rigid universal logic. Each contact is different, and the company's business dealings with each are unique and ever-changing. The communications between the company and the contact cannot be determined by an isolated LOB application that knows nothing about other areas of the business.

- *Rationalizing discrete systems*—Although common applications such as calendars and e-mail will remain, we need to design a way of pulling together all relevant data into a single composite engine and properly prioritizing to-do activities. This means the system needs to process tasks, make requests, and obtain commitments, provide reminders, follow-ups, and feedback, and make this information accessible to everyone in the enterprise by using a consistent user interface. And as Stephen Covey suggests in his book *Seven Habits of Highly Effective People*, we need to move beyond the concept of priority as a single value and deal with both importance and urgency.[4]

- *Handling informal workflow*—The user should be able to trigger automated activity sequences and alter them on the fly when appropriate. Workflow software has historically concentrated on situations where workers are dedicated to one major process, such as processing insurance claims. In every office, however, many informal or *ad hoc* processes need to be supported as well. For example, an overdue payment could trigger an automatic collection call or possibly a series of calls and faxes, all of which need to be recorded. On other occasions, a user may take a call and need direct access to the proper record in the LOB application for a shipping inquiry or for order entry.

- *Managing workloads*—A kind of *intelligent gating* must be provided. Once the user's workload is known, the system needs to manage the input so that the workload remains manageable and prioritized. Every company has specific persons who know unique and valuable things about the business, and to whom we go when we need information. If their telephone is busy or their door is closed, we have to go elsewhere. This means the busy phone signal or the closed door serve as gating factors. In an electronic system, similar structures are needed to manage workloads and prevent a disillusioned and overworked user from simply "pulling the plug" and working outside the system.

Managing informal data

The creation of business frameworks dealing with knowledge information that can be used as a foundation for developing applications will serve to facilitate the integration of the informal pieces of knowledge with the formalized information provided by the LOB applications. To develop these frameworks, we first need to identify the structures that are the components of this informal knowledge sharing.

Three basic structures, contacts, activities, and topics, appear to provide the necessary elements for recording and sharing knowledge. As shown in Figure 1, each one of these structures can be composed of two kinds of data: structured and unstructured. Structured data are those attributes that are fixed and predetermined, such as address, telephone number, date of creation, author of an activity, and description of a topic. Unstructured data are notes that record, in the user's own words, information about each object. It is in these unstructured data that tacit knowledge is exchanged and recorded.

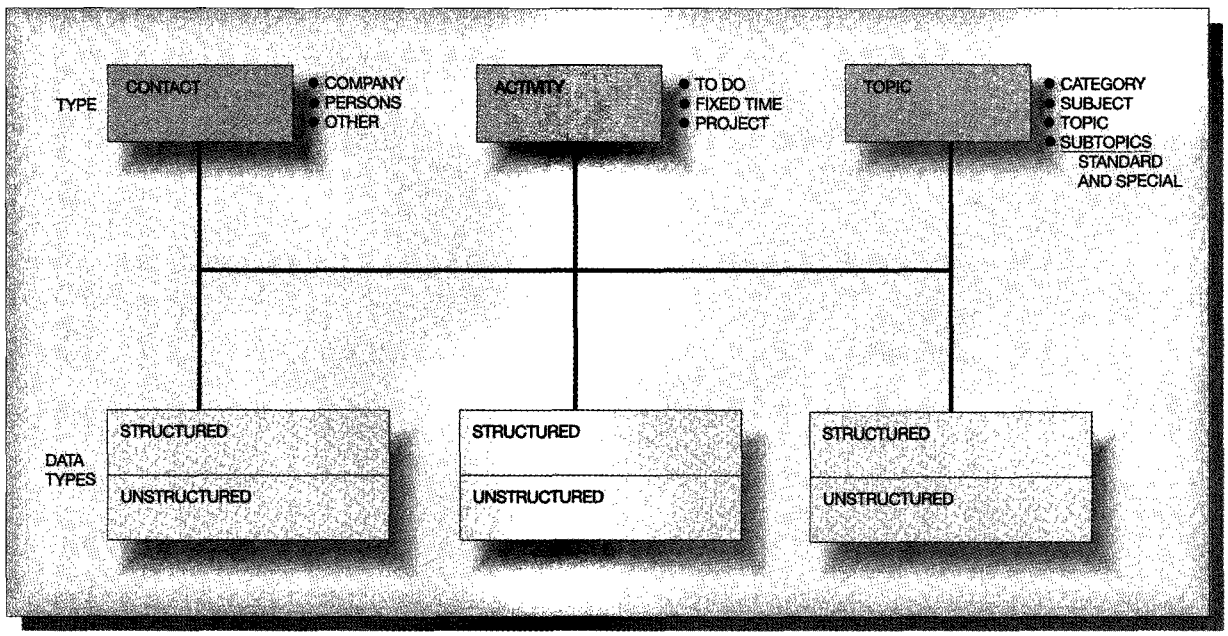
Contacts. Contact data are profile data on companies and people. Such data include customers, prospects, employees, and vendors—in fact, every person or company with whom the company does business. Although the data change over time, they are not time-dependent.

LOB systems keep information about contacts on their customer master files, their vendor master files, and their employee master files. Contact management applications keep this information in address books and directories. The often-quoted reason that one single place for information about our contacts cannot be used is that each application requires different information, and it cannot be predicted what information the next application will need.

But the persons whom we call on the phone may work for companies that owe us money, or have placed an order, or supply the parts we need for manufacturing. When the address of the contact is changed, it should be changed in one place only. The contact should "know" its identity in LOB applications, so that links can be immediate. It is the same contact. General, nontransactional information for a contact should be accessed by all applications from the same source.

If the contact is to serve as a universal source of information about a company's vendors, employees,

Figure 1 Workplace automation core objects (patent pending)



customers, and prospects, it must be easy to change or add to the information it contains.

One of the principal requirements of a contact framework is that a mechanism for changing the contact's coded information must exist. It is not sufficient to provide a configuration tool that can be used by a technical person to configure the information to be stored for each company. Some companies do not have a technical person to perform this type of job. In addition, the needs of each company are dynamic and varied, requiring different pieces of information to be kept at different times. Our experience has shown that many people want to reconfigure their contact information as often as twice a month. For this reason, the mechanism to reconfigure the information in a contact must be something that is accessible to the end user.

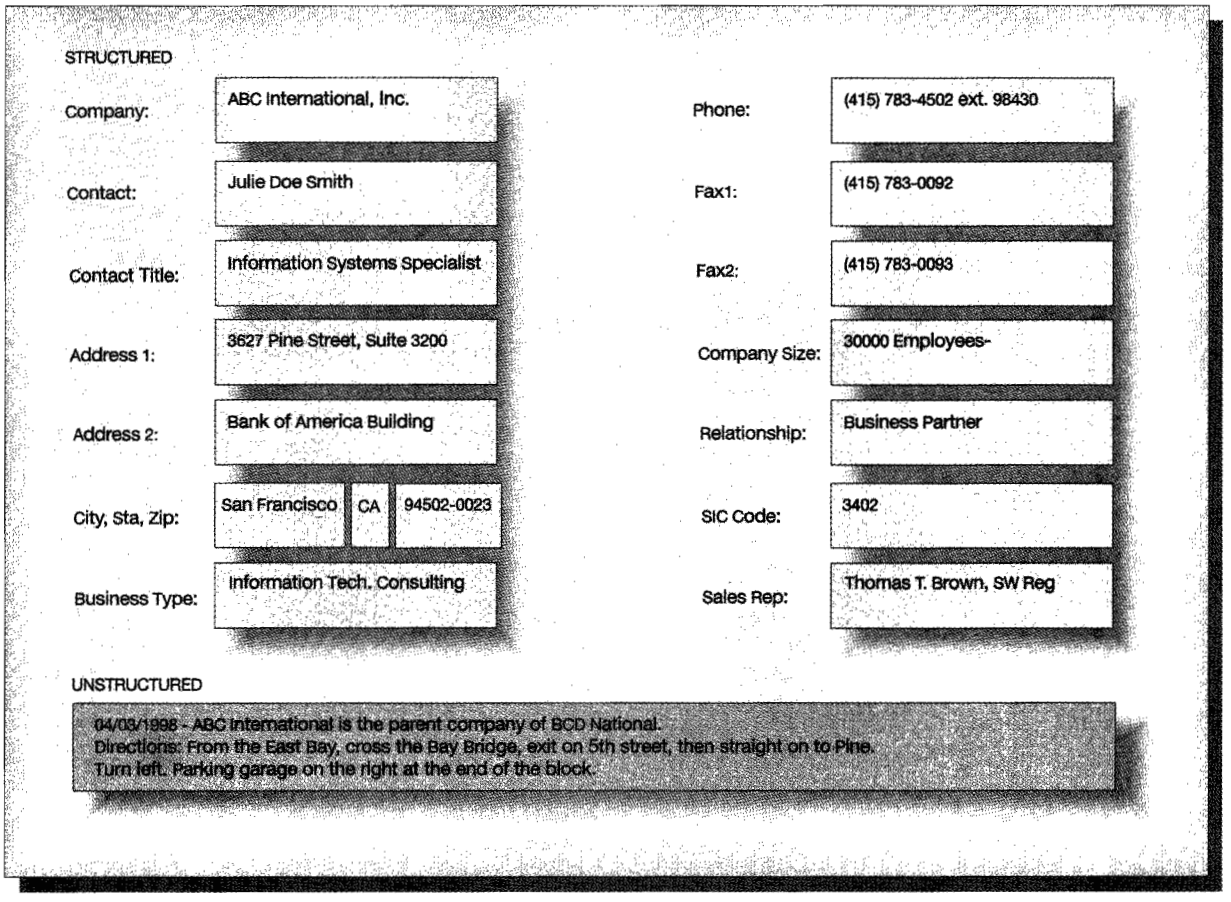
In addition to the structured information, notes need to be kept about a contact. As Figure 2 shows, the note information is as important as the structured information. It is important to know that this company is the parent of another with which we may have had a past business relationship, and directions are always useful.

Activities. Activities are the second of the three structures.

- They can be used to organize the workload of all the users of the application.
- They can be used as a means of communicating among users.
- They can serve as documentation of the communications between a company and its contacts.

Activities are meant to document and enforce the request or commitment mechanism by which work is accomplished. Howard Goldman of Management Associates defines the communications among employees in a company and between employees of the company and their outside contacts as a series of requests and commitments. The quality of each commitment determines the quality of the work produced by the company and the image that it projects. Each request is asking for a commitment from someone to do something by a certain date. Based on his or her confidence that the commitment will be made, the requester will make further commitments to others. When a commitment is not met, a whole chain of commitments is placed in jeopardy. In an organization, many informal commitments are made that

Figure 2 Contact data



cannot be kept because there is no formal recording, and it is easy to forget promises that have been made, especially when the due date is several weeks away. It is also easy to over-commit when the person cannot view all prior commitments at the time of a new request.

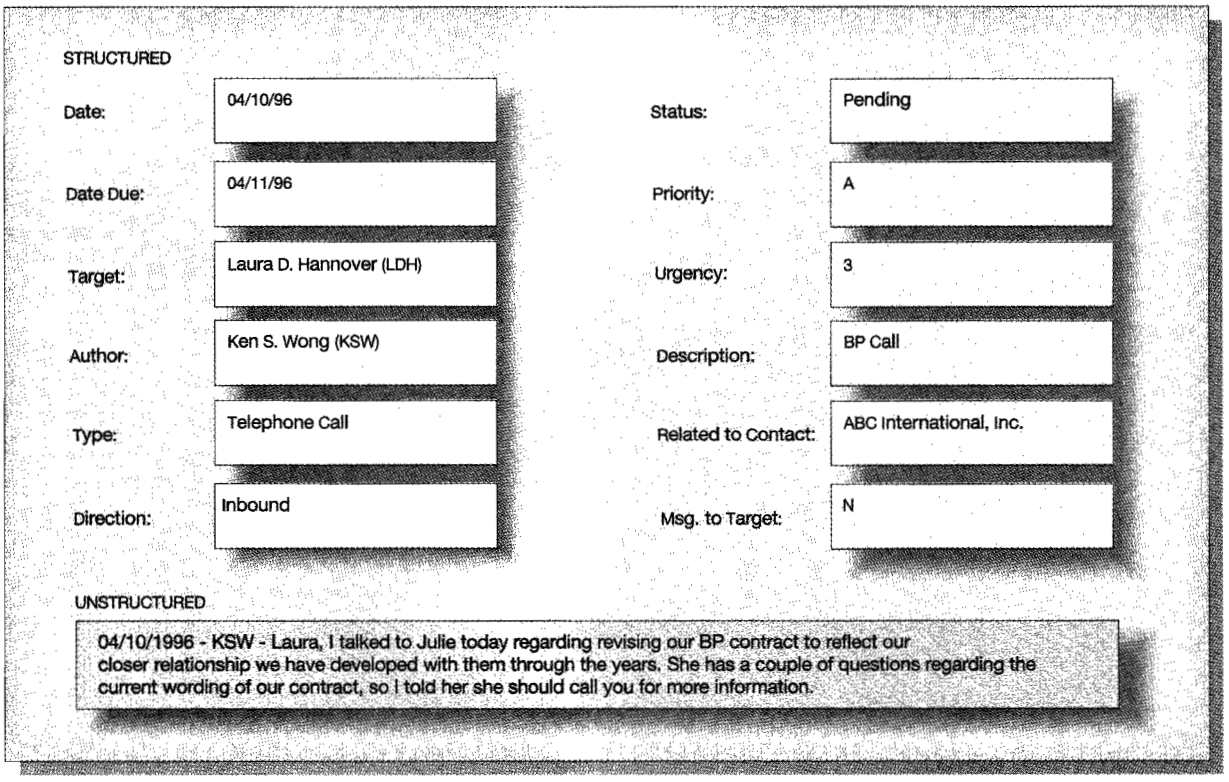
An activity can serve as the mechanism to record the request and the date by which the work requested needs to be completed. The user of whom the work is being requested can use the same mechanism to refuse the request, or to commit to performing the work by a date called the commitment date, which is usually the same as the due date, but at times is not. A start date can also be defined for those activities that cannot be completed in one day. Each user can have a to-do list that shows all activities to be performed that day, from phone calls to be made and meetings to attend, to ongoing project work.

Activity data, as opposed to contact data, are time-dependent. As Figure 3 shows, the structured data contain the information about the activity, such as date, time, type, and contact names, author, and target, which is the person performing the activity. Unstructured data in the form of a note are also a part of an activity. Unstructured data can record the emotional content of the request, explaining the urgency of a task to be performed, or recording the tone of a telephone conversation.

We have identified at least three kinds of activities:

1. *To-do activities*: These are short-term tasks that must be completed before a certain date but are of short duration and do not have to be performed at a specific time, such as telephone calls, faxes, e-mail, or documents of any type created, received, or sent.

Figure 3 To-do activity



2. *Fixed-time activities*: These are calendar events that must be performed at a specific time on a specific date, such as meetings, conventions, and other events that have a planned date, time, and duration.
3. *Project activities*: These are longer-term tasks, such as projects, where the action can span days, weeks, or even months, and where the activity is dependent on the completion of other activities or is a requirement for other activities.

Although some existing applications handle one type of activity, a framework is needed that will deal with all kinds of activities, providing the user with one consistent interface. At present, all the user has available are unconnected applications. Examples of such applications are calendar systems that do not handle to-do or project activities well, tickler files that do not handle fixed-time or project tasks well, and project management systems that do not handle to-do or fixed-time tasks well.

Each activity should be directed at only one person, specifically the user who is being requested to perform the activity. In some cases, more than one activity has to be created because more than one person is involved. It may seem that in these cases the activity should be directed at more than one person, but the request or commitment process works better when multiple activities are created, one for each person.

For example, you may want to inform three persons about the outcome of a meeting with a contact. If we have only one activity for these three persons, the activity would appear on the to-do list of each of them. But the commitment mechanism could not be enforced. A request being made to each person may be different, and the due dates and commitment dates may be different as well. The best way to handle this situation is to treat each request as a separate activity, each directed at a different person, and each with its separate due and commitment dates.

However, the note attached is the same note for all three activities. In this case, the three activities form a subtopic.

Subtopics. A *subtopic* is a collection of one or more activities that share a note or group of notes. The following is an example of a subtopic.

A user named Eva received a call from a vendor with a special offer. She wants to inform her supervisor John about the offer, and then pass the issue to a colleague, Roger, who will call the vendor back with an answer. This procedure could be handled by phone and voice mail systems, a series of e-mail messages, or Post-it** notes, but none of them serves completely. For example, where do Eva's requests fit into the existing priorities for John and Roger? How does Eva ensure that her requests were handled correctly by John and Roger? And how do we get a permanent record of what actually happened?

Consider a better alternative. Eva takes the call and creates an activity relating to the vendor's contact. She types a note for John's benefit describing the offer and includes a question to Roger asking him to call back once John signals his appraisal. She adds a pending activity in the new topic for John today and for Roger tomorrow. The activities will appear in John's and Roger's to-do list correctly organized by priority. If Eva chooses, she can set a grace period for both activities that ensures she will be notified if individual actions of Roger or John do not take place in time.

Both John and Roger are notified of Eva's request. The request is different for each. In John's case, he is simply asked to read the note and indicate that he has done so by completing the activity. In Roger's case, Eva is asking him to respond to the contact's question. She has made the due date different for each activity because she does not want Roger to call the contact before John has had a chance to view the information. Once Roger reviews the activity, he may find that the answer requested involves research that cannot take place before the due date. In this case, Roger sets the commitment date to the date when he can send the information to the contact, and he will add to the note indicating that he cannot commit by the due date. He then completes the activity and creates a response activity to Eva.

All three activities described here belong to the same subtopic because they share the same note or group of notes. There may be other subtopics for this same contact, for example, conversations about a price negotiation or telephone calls made to track down shipments. After a while it may become difficult to find which subtopics relate to the same issue, especially after some time has passed and numerous other conversations, requests, and commitments have been recorded, all related to this vendor's special offer. A way to group subtopics that relate to the same thing is to put them in a topic.

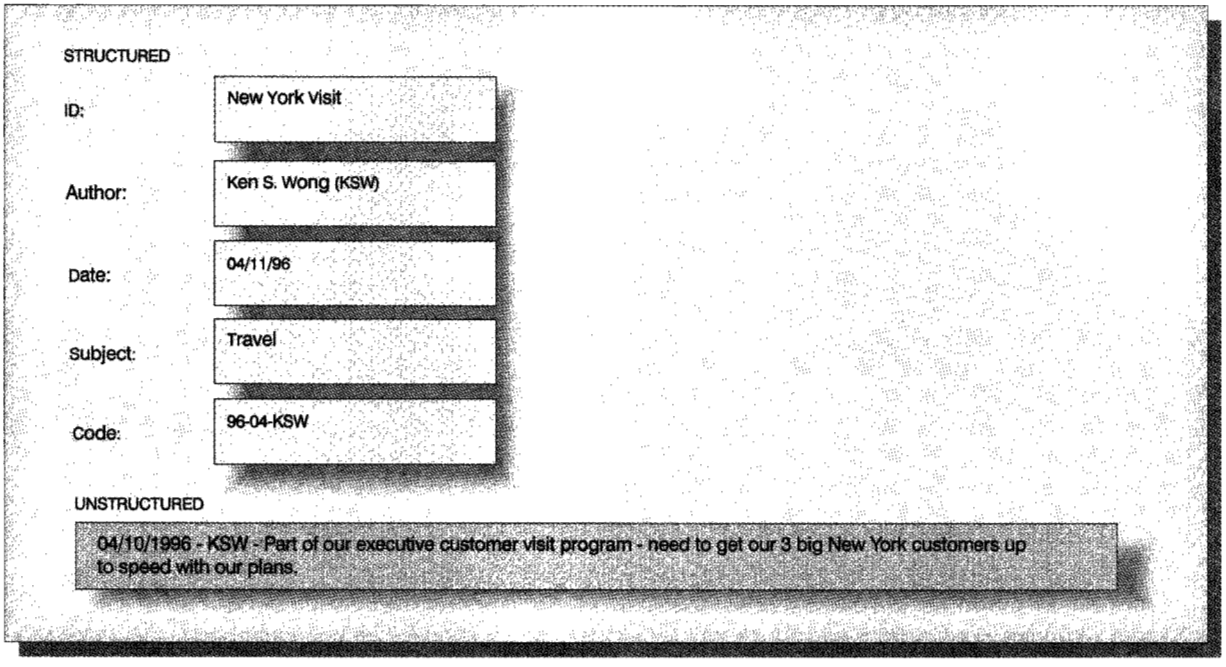
Topics. *Topics* are collections of subtopics that have been grouped together by a user who has determined that they have a common theme.

Figure 4 shows that topics could exist within a hierarchy of levels that describe the category, subject, and other attributes. Subtopics may exist in more than one topic. In the example above, the subtopic logically belongs to the topic of "Activities Relating to This Contact," but the user may also want to create a special topic for "Special Offer" and include this subtopic there, along with others that may be created over time.

An example of a subtopic included in more than one topic is shown in Figure 5. In this case, all activities concerning the sale of an upgrade to a customer are in a topic called "Selling Upgrade to Smith & Company." This topic may include all phone calls, letters, faxes, and internal communications related to this sale. It may also include the contract to be negotiated in person. The salesperson may be planning a trip to New York, and may create a topic called "New York Visit." In this topic he or she may include all the arrangements being made for the trip, and some activities concerning visits to customers in the area, including the contract to be discussed with Smith & Company. In this way, the salesperson has grouped in one place everything that has to be done or has been done concerning the New York visit. In the future this information can be found and viewed in the context of all our business dealings with Smith & Company by using the default topic of all activities for the contact, or we can see all the activities having to do with this particular sale by viewing the topic labeled "Selling Upgrade to Smith & Company," or we can view the results of the salesperson's visit to New York by viewing the topic labeled "New York Visit."

If topics are to serve as logical organizers for activities, the user needs to view any activity by displaying a topic that contains it. The user can change the state of the activity from any topic display. The

Figure 4 Topic data



change should be immediately visible to all other users who may be viewing other topics containing the activity.

When you consider that any type of activity and activity notes can be included in a topic, this becomes a powerful concept. For example, many things that discrete systems handle are in fact topics. The problem with these discrete systems is that these topics are currently maintained in separate data stores and accessed by various users through different user interfaces that can deny effective access to some users.

In addition, though we are not used to thinking of everyday knowledge worker tasks as being in a topic, all tasks are part of the stream or sequence of activities that users may group into *ad hoc* topics. Although this sounds like extra work, it is simply giving the user the mechanism to record what is already known—the benefit being that once recorded, retrieval at any future time is simple.

Organizing work

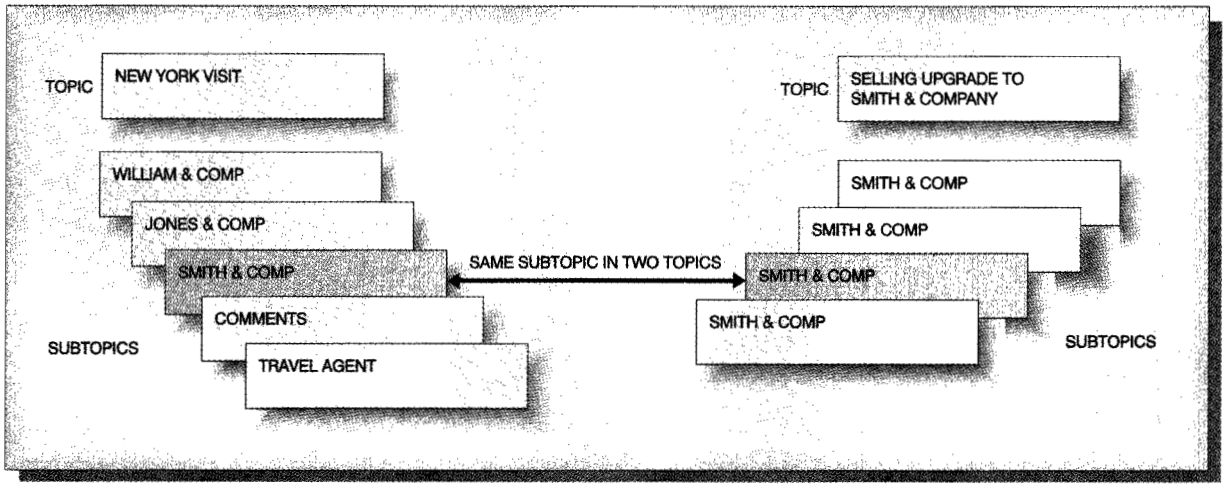
Topics and subtopics serve as means to organize and group activities so that they can be easily retrieved in logical groupings. The power of organizing work, however, is in the activity itself.

The user needs to be able to schedule activities for her- or himself and others, view activities that have been scheduled for her or him in priority and date order, and monitor activities that she or he has authored.

Each activity has an author, who is the requester, a target, who is the person who will make the commitment, a due date, a committed date, and each activity usually relates to a contact on behalf of whom the activity will be performed. The target of the activity can be the author as well. For example, I might want to set an activity to remind myself to write a letter to a customer two days before a road show, reminding the customer to attend.

Activities also have priorities. The priority may be determined by the author of the activity and should have two parts, the activity's importance and its urgency. Not all that is important is urgent, and not all that is urgent is important. For example, getting a contract to a customer by the end of next week is highly important but not urgent. Returning the call

Figure 5 Topic organization



of a friend who wants to have lunch with you may be urgent but is not very important to the company. By allowing the requester to make this distinction, the target is provided with information that will enable him or her to determine which requests must be honored first. It allows all users in the company to keep the important requests in sight instead of losing them in the chaos created by urgent, perhaps not important, requests.

The completion of an activity often signals the end of one step in a business process and the beginning of another. A next-step option creates follow-up activities that are based on the results of the completed activity. This is a way of further helping users organize their workloads.

For example, let us say a user operating as a telemarketer found an interested contact. The user should create three separate activities: one to schedule literature to be sent to the contact immediately, one to notify a salesperson to call in four days and make sure the literature arrived, and one to remind the user to follow up with a phone call in two weeks to make sure the salesperson called back.

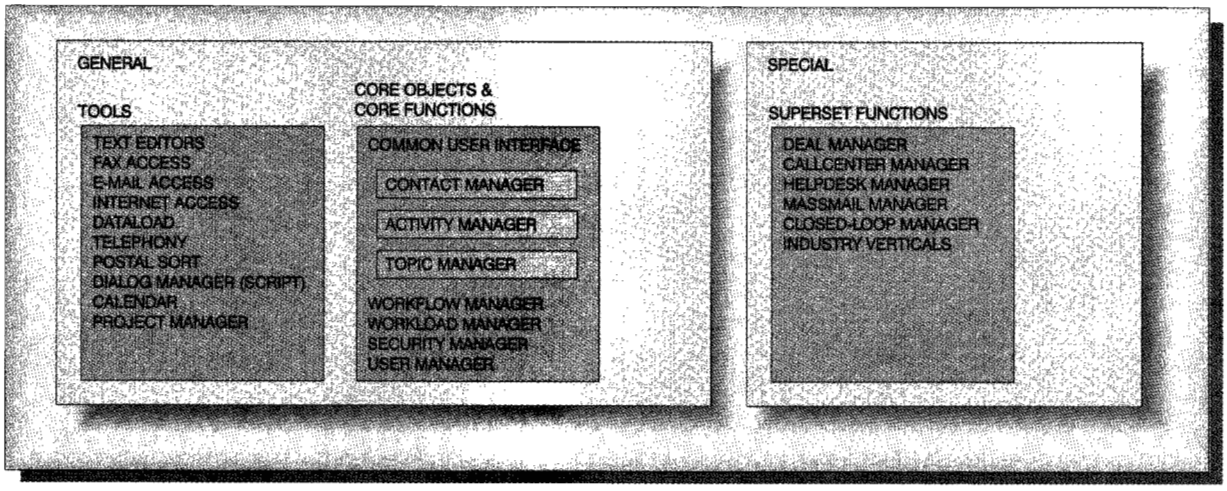
We can rely on the user to create these activities for the correct target, but chances are the user will forget one or all of these activities. Instead, a set of result codes can be defined with a next-action sequence. The user will then key in a result code, and based on what is keyed, a future activity or a predefined set of activities can be created. These activities may have different targets and will appear on the to-do lists of each of the targets at the appropriate date and time. An administrator should be able to set up both the result codes and the activities they will trigger.

This procedure in effect forms the basis for a dynamic workflow system, where the next step in the flow of information is determined by the user either consciously by creating activities for others, or in an automatic fashion by triggering one of a set of next-step events based on the completion of the activity.

Communicating among users. Activities are often viewed as individual to-do items, with only one user participating in their performance. In fact, many contact and activity application packages provide the user with a simple to-do list and a calendar, but that is all. Communications in a company are much more complex than that. When we consider enterprise-wide communications, we need to group activities that allow users to carry on a dialog for all to see.

In publishing communications that take place within an enterprise, many people think of e-mail. But e-mail relies on a person's knowledge of the use this information may have, his or her ability to define all the people who may need access to it, and his or her willingness to direct the mail to these people.

Figure 6 Workplace automation functions



The subtopic and topic structures present a user with a better alternative to internal e-mail. The information is not passed to someone and left sitting in a mailbox. It is immediately available to all interested and authorized parties, not just to the targets of the activity groups. It is available in its entirety, with all conversations recorded, and all requests and commitments fully documented, not in pieces. It is available for future reference. And there is only one version of the information that all can see.

Letters, faxes, visits, e-mail, and telephone calls are all a means of communicating with a contact. Topics serve to group these activities for easy retrieval. By creating an activity framework that can be used by application developers, we can ensure that all communications between the company and its contacts are documented for future use.

Use of San Francisco frameworks

Some of the concepts that were described have formed a part of our application, ActionWare/400**, for some time, but our research into the domain of knowledge applications has prompted us to search for ways to create the three business frameworks as independent entities that can be used as a base for developing other applications.

Because knowledge applications require a good graphical user interface and links to existing desktop applications, as well as links to server line-of-business applications, a client/server system using object-oriented programming is the right way to code these frameworks.

The San Francisco initiative presents us with the opportunity to create these frameworks both for our own use in our contact and activity management application, and for the use of others developing other knowledge or line-of-business applications.

The base San Francisco frameworks provide the basic functions that take care of security, concurrency, and persistence and distribute object management. Upon these base frameworks, the business frameworks of contact, activity, and topic can be built and made available to developers working on multiple platforms.

Figure 6 shows the set of applications that can be built over these business frameworks.

The contact frameworks would provide a user-configurable contact management system. User-defined algorithms can create multiple access keys based on input the user is required to key in to other fields. Several search and navigation mechanisms would be provided, as well as the ability to define restricted information and indicate which users will be allowed to access it.

The activity frameworks would provide an activity management system with the ability to group activ-

ities into subtopics. The basic activity types would be handled (information, action, fixed time, and project), as well as the basic methods of distribution (internal, telephone, e-mail, and fax). The developer would be able to extend the framework by creating new types and methods.

The framework would work with both aspects of priority, importance and urgency, and a next-action mechanism would be provided to allow the user to create a dynamic workflow. The application developer would have extension points to modify and add to the framework.

The topics frameworks would allow the user to add subtopics to topics and action activities from different topics, and to search for topics based on keys generated by algorithms that developers can modify.

Summary

Since the advent of the punched card, we have progressively computerized and centralized the core business accounting functions, but we have left the recording of human interactions in documents and notes to the individual user to handle with paper or PCs.

In the 1990s, these informal data have now been recognized as the key to business knowledge sharing, but the systems in which these data are stored are fragmented and dispersed among users.

If we critically examine how we might best make a major contribution to both the personal and group productivity of knowledge workers, the need to bring order to knowledge applications and to re-establish the natural linkages that should exist to LOB applications becomes obvious. However, existing systems and people habits represent massive inertia, so the opportunity to make real changes comes along but rarely. The San Francisco project represents such an opportunity. The requirement is to get the basics right, which will allow developers to easily deliver naturally integrated systems.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of 3M Company or ActionWare.

Cited references

1. K. Bohrer, "Architecture of the San Francisco Frameworks," *IBM Systems Journal* **37**, No. 2, 156–169 (1998, this issue). Also see http://www.ibm.com/Java/Sanfrancisco/, IBM Corporation, Network Computing, Armonk, NY.
2. Press release, Buckman Laboratories, 1256 W. McLean Boulevard, Memphis, TN 38108 (1996).
3. I. Nonaka, "The Knowledge-Creating Company," *Harvard Business Review* **69**, No. 6, 96–109 (November/December 1991).
4. S. Covey, *Seven Habits of Highly Effective People*, Simon & Schuster, New York (1990).

General references

H. W. Chesbrough and D. J. Teece, "When Is Virtual Virtuous?" *Harvard Business Review* **74**, No. 1, 65–71 (January/February 1996).

"Making Companies Efficient," *The Economist Newspaper Limited* (December 1996).

K. Stephenson and S. H. Haeckel, "Making a Virtual Organization Work," *Focus*, the Zurich Customer Magazine, No. 21, 26–30 (1997).

Accepted for publication December 18, 1997.

Robin Retallick *ActionWare, 5801 Christie Avenue, Suite 500, Emeryville, California 94608-1928 (electronic mail: rretallick @actionware.com).*

Mr. Retallick is President and CEO of ActionWare. A graduate electrical engineer from the South Australian Institute of Technology, he left the engineering profession to join IBM Australia in 1970. He became a highly awarded and top-performing branch manager who spearheaded the implementation of new CEO education and strategic planning methods. He spent three years with IBM in New York in a variety of marketing positions related to the IBM System/38™, and returned to IBM Australia in 1982 to manage product marketing. In 1983, he returned to the United States as President of ActionWare. The company's product, also called ActionWare, was first produced in April 1989, designed specifically for the IBM AS/400. ActionWare was a winner of the 1996 Arthur Andersen Best Practices Award in the category of knowledge sharing. Mr. Retallick has been active in the design of state-of-the-art workplace automation systems and is a sought-after speaker and an author of many articles. He has been an active participant in the San Francisco project Advisory Board and Reference Group since 1995.

Susana Sánchez *ActionWare, 5801 Christie Avenue, Suite 500, Emeryville, California 94608-1928 (electronic mail: ssanchez @actionware.com).*

Ms. Sánchez is currently Director of Frameworks at ActionWare. She received a chemical engineering degree from Universidad de Nuevo Leon and an M.A. from the University of San Francisco. She has 20 years experience in line-of-business applications, especially manufacturing and distribution, and is APICS-certified in production and inventory management. After 10 years with IBM as a systems engineer and systems engineering manager, she joined ActionWare where she focused on developing contact and activity management applications. Ms. Sánchez has been an active participant in the San Francisco project Advisory Board and Reference Group since 1995.

Reprint Order No. G321-5672.