


Activity management as a Web service



A. Cozzi
S. Farrell
T. Lau
B. A. Smith
C. Drews
J. Lin
B. Stachel
T. P. Moran

In this paper, we present a new method for organizing collaborative work. This method is based on the concept of “activities,” defined here as high-level structured representations of the people, artifacts, and processes involved in work and their relationships. We show how users and developers can leverage this representation to enhance productivity, collaboration, and business applications. Central to our vision is an interface to activity data which is lightweight, based on Web Services, and enables activities to be easily integrated into the applications and tools people already use. We describe the Wax system for activity management, which implements our model of unified activity using both semantic Web and REST/XML (Representational State Transfer/Extensible Markup Language) approaches. We describe several user interfaces that let users interact with activity data, and we discuss our experiences using the Wax system for two case studies that involve coordinating a large event and managing accommodations for new employees.

INTRODUCTION

This paper is a companion to “Business activity patterns: A new model for collaborative business applications,” which also appears in this issue of the *IBM Systems Journal*.¹

Today’s tools provide little support for team members working together on a collaborative process. E-mail is the predominant communication tool used today, and it has been overused for purposes other than simple communication, such as exchanging files, scheduling meetings, and archiving data.² Using e-mail to manage activities has many drawbacks. For example, it can be difficult to determine the current status of an activity which is managed by e-mail, and if people join an ongoing

activity, it can be difficult to bring them up to speed with other team members.

At the other end of the spectrum are formal business-process-workflow systems. These systems direct processes and the people involved in them, but are overly rigid for most everyday business activities.^{3,4} A middle ground between e-mail and workflow systems would better suit many collaborative activities.

©Copyright 2006 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the Journal reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to republish any other portion of the paper must be obtained from the Editor. 0018-8670/06/\$5.00 © 2006 IBM

The goal of the Unified Activity Management (UAM) project at IBM Research is to define a new model for collaborative work based on a shared semantic representation of collaborative activities.⁵⁻⁷ “Activities” as used here refers to a digital schema-based representation that describes the properties of a collaborative work project and semantically relates the people, artifacts, tools, events, and other elements which are involved in carrying out the project. Examples of activities include organizing a large event or conference, responding to a request for proposals, and resolving a trouble ticket (mechanism used in an organization to detect, report, and resolve a problem). The activity model and how it is used to support business applications is described in depth in a companion paper in this issue.¹

The UAM approach

The objective of the UAM project⁸ is to design a system that supports collaborative work processes, with multiple people coordinating their work in order to accomplish a shared goal. Our work is based on the assumption that there is a great potential benefit in supporting the non-structured aspects of everyday business activities, those that are not managed by workflow processes and existing corporate applications. These kinds of activities are often managed by using handwritten notes, e-mail, telephone conversations, and other informal means. This objective has led to a number of choices in how activities are represented.

First, we believe that activity representations should have semantics and structure. For instance, each activity has a creator, a title, a description, and a set of people involved in its execution, each with a potentially different role (participant, observer, etc.). Activities may have resources associated with them, such as Web pages or word-processing documents; resources may be of different types, such as a reference document or an output of the activity. We hypothesize that formalizing the activity structure explicitly enables the participants in the activity to see how the different parts relate to each other and to more easily track the current status of the activity. In the section “Unified Activity ontology,” we describe our representation of activities.

Second, we believe that activities are fundamentally composed of metadata, as opposed to content. Activities serve as the glue that joins individual items of content created and managed in word

processors, spreadsheets, e-mail, and Web applications. Rather than reinventing each of these business applications in a new, monolithic application, we take the position that activities should provide a framework for collecting all of these items and presenting them in a single, unified view. As a result, we have developed a model that we call “activities as service”: a lightweight Web service infrastructure for creating, managing, and querying activity data. We have used this infrastructure to develop Web-based activity management systems. More important, however, we believe that activity data is most useful when presented within the context of the tools and applications people already use.

This paper describes our representation of activities and presents the Wax system, a Web service framework for activities that leverages a semantic representation of activity. Wax takes advantage of emerging technologies such as lightweight (REST [Representational State Transfer]) Web services, RSS (Rich Site Summary), and the semantic Web to provide access to activity-related data as a service. We present the results we obtained in using the Wax system to manage two large business activities and discuss which features of our design were most helpful to the participants as they used the system.

Related work

Previous approaches to supporting collaborative tasks generally fall within the categories of workflow systems or personal information managers (PIMs).

Formal workflow systems are often rigid and frequently assume fixed roles for users and a fixed pattern for actions. One such system is the Coordinator.⁹ These systems are characterized by a rigid specification of the processes to be executed. Furthermore, workflows tend to work as independent entities, having little integration with the rest of the computing environment. A more flexible workflow is described in Reference 10, wherein end users can modify the process. Our system goes even further by dispensing with the process model altogether.

The Task Manager¹¹ is the earliest system of which we are aware that is based on shared representation of tasks that are malleable and that relate people and resources. A later system that is even closer to our

approach in using an early semantic network representation is described in Reference 12.

Shared workspaces provide shared access to documents (such as the Groove system¹³ and Lotus* Notes* TeamRooms). These systems tend to be difficult to use for simple, lightweight activities, and it is unclear how they might integrate ad hoc activity with more formal business processes or workflows.

PIMs aim at improving personal productivity by organizing communications, contacts, and events related to an individual. They do not support shared entities, and external interaction is handled through messaging. In contrast, our system is centered around activities and uses them to organize documents, people, and events.

More details about the integration of our system with business processes are described in Reference 1 and Reference 4.

The remainder of this paper is structured as follows. We begin by introducing a semantic representation of activity, based on the Resource Description Framework (RDF), and we describe the ontology used to represent activities and their properties. We then present the Web service APIs (application programming interfaces) that we have defined to provide access to activity data from Web applications and third-party extensions. Next, we present the user interfaces and client plug-ins that we have developed, which let users interact with activities. Finally, we report on the results of two case studies in which the Wax system was used. Our results indicate that the participants found having an activity management system to be extremely useful and confirm our hypothesis that a structured activity representation brings value to activity management. We conclude with a discussion of directions for future work.

EXPLICITLY REPRESENTING ACTIVITIES

The goal of activity management is to help users be more productive by organizing the work they do around the concept of activities. In order to help users manage activities, they must be represented in a consistent way. This representation should capture the essential semantics of an activity: the links, relationships, and resources that differentiate it from other activities.

It is important to distinguish between the typical representation of real-world activities in the minds

of the people involved and explicit activity representations. Real-world activities are often implicit (or tacit); people simply perform activities without any representation of them. Real-world activities can also be deliberately driven toward a more or less well-articulated objective, as proposed by Activity Theory.¹⁴ In contrast, real-world activities can also have explicit representations, such as activity descriptions in some medium, for example a plan written on a whiteboard. We propose that explicit computational representations of activities (i.e., representations enabling an activity to be processed with computational tools) are useful for managing them. Explicit representations can be more or less elaborate; it is our intention to support fluid transitions between various levels of elaboration, based on people's estimates of the costs and benefits of creating them.

Explicit activity representations can be formal or informal. Informal representations place no constraints on how the activity is represented; it may be written down as a textual description or may consist of scribbles on a Post-It** note. The goal of our work is to provide a unified activity representation, which captures the common properties of activities in a standardized representation so that activities can be shared and managed by different systems. In order to achieve this goal, we require activity representations to follow a formal vocabulary, which captures the common characteristics of the activity in a unified representation so that it can be processed with computational tools.

In analyzing real world activities, we found a large amount of variability in what is needed to represent different kinds of activities. For example, an independent consultant may want each of his activities to include a property denoting the client for which the work is being done and the billing rate for each client. On the other hand, a programmer might want to annotate each activity with a defect report number and the sections of source code that are relevant to the activity. As a result, activities should be represented as objects with a large number of optional fields that cannot be predicted. This has several important implications for the design of the API and the data model.

Increasing explicitness and formality puts a burden on the user that must be counterbalanced by some expected payoff. The first incentive to move to explicit representation is sharing: a representation of

an activity becomes a communication artifact shared by more than one person or group. The second incentive is the automatic support provided by the system. Our activity system strives to attain a principle of incremental benefit where the more the user invests in constructing a formal representation of an activity, the more support that user can get from the system, and this benefit is commensurate with the additional effort invested by the user.

Activity representations in the mind of the user do not require any support, and informal representations are not particularly problematic; but the formal representation of activities is quite a challenging problem. If we truly want to encompass the breadth of human activities with an activity management tool, it must deal with a difficult representation problem. Different kinds of activities should be represented in a common “language” when possible, but the representation must be extensible to support new types of activities and functionalities. It is not desirable that the user be required to specify an activity “type” in advance—the user may not have decided on an activity type, and the activity may change considerably during its lifetime; for example, starting as an e-mail or a “to do” entry and evolving into a multi-person project. What is needed is a fluid representation that enables activities to change and acquire new properties.

The representation problem is further complicated by the requirement to represent not just activities, but potentially all kinds of domain-specific objects related to them (people, documents, files, calendar appointments, Web sites, orders, etc.). We came to realize that the network of relationships that binds other objects to an activity is one of the most important features of an activity. We could use URLs (Uniform Resource Locators) or other identifiers to represent these objects, but then we would be severely limited in what we could do with them or the kind of queries that would be possible. Ideally, we would like to associate selected metadata with those objects in order to support queries within the activity system, for example, displaying all activities with a calendar appointment occurring today. The challenge of the task is to represent highly variable objects and their relationships to potentially any other object.

We have investigated the use of semantic Web¹⁵ technologies to provide a consistent, standards-based environment for the representation of activ-

ities. RDF,¹⁶ a key component of the semantic Web, provides a representation flexible enough to support our generalization of activities by enabling relationships from multiple ontologies and data sources to be viewed as one coherent data structure: namely, a graph. RDF comes with a data model to express binary relationships, query languages (RDQL¹⁷ and SPARQL¹⁸), several implementations of storage repositories, a file format (RDF/XML), an ontology language (OWL¹⁹), and an inference model. As part of our investigation, we have developed a unified activity ontology, based on OWL, that defines the generic concept of collaborative activity and provides a common set of properties to ensure a level of consistency and uniformity for all types of activities. This ontology can be easily extended to new relationships and properties, thus enabling activities to be customized for particular end users without necessitating a system redesign.

Unified Activity ontology

The Unified Activity ontology defines a few fundamental objects: (1) activities, (2) actors (people or software agents), (3) events (calendar entries), and (4) resources (files and URLs). The ontology builds on the Dublin core,²⁰ Friend-Of-A-Friend,²¹ and iCalendar²² ontologies to describe standardized properties such as titles, descriptions, and e-mail addresses.

Table 1 summarizes the key properties of the Unified Activity ontology. The basic entity is the activity, characterized by a few descriptive properties (title, description, result) and several relational properties that connect this activity to other objects, such as actors and resources.

An *actor* represents a person in the activity system. The basic description of an actor includes a name and an e-mail address. An actor must be involved in an activity with a particular role; we have defined several roles, such as participant, observer, committed, and doer.²³

Events represent the time-centric features of an activity. We use the iCalendar standard to represent properties of events, such as the title, description, and start and end dates.

Activities may include *resources*, which represent external artifacts (Web pages, files) that are related to the activity. Each resource is described by a

unique identifier (URI [Uniform Resource Identifier]) and a label. Resources may be related to activities in a variety of roles; for example, a document may be a reference document (which is consumed in the process of completing the activity) or an output document (which is produced as a result of the activity).

Activities may be decomposed into *subactivities*, which are activities on which the “parent” activity depends in some way. The subactivity relation provides a way to organize the structure of an activity and define the breakdown of the work involved in completing it. In keeping with the RDF graph model, the subactivity relationship is one that links a parent activity and a child activity. Because the child activity is not contained in a single parent activity, an activity can be a child to multiple parent activities.

We also support the concept of an *activity pattern*, which is a special type of activity that is designed for reuse. Patterns are well-suited for capturing the best practices for conducting an activity; if the activity needs to be repeated, one can create an instance of the pattern and customize it for the new activity. Examples of activity patterns include planning a meeting, running a software project, and hiring a new employee. Patterns are explained more in the companion paper by Moody et al.,¹ and one of the case studies later in this paper illustrates the use of an activity pattern.

RDF as a development environment

Our RDF environment uses the Jena toolkit from HP Labs. Licensed under an Apache-like open-source license, our project utilizes an RDF API, SPARQL and RDQL query processors, an HTTP (Hypertext Transfer Protocol) RDF API known as Joseki, and a graph API for persistence. We have augmented the Jena toolkit with a JSP** (Java ServerPages**) ²⁴ tag library that facilitates Web development, and we have augmented the graph API to enable integration with remote data sources at the RDF level.

Activity data stored as RDF can be accessed through the JSP tag library by using constructs such as:

```
<rdf:resource id="focus">
  Title: <rdf:property name="dc:title"/>
</rdf:resource>
```

which retrieves the Dublin core “title” property from a node identified as “focus” and prints it in a

Table 1 Unified Activity ontology in the Wax system

Subject	Relation	Object Description
Activity	URI	Unique identifier for this activity
	Shortname	Short human-readable identifier for the activity, such as an e-mail name
	Title	Short description of activity
	Description	Longer description of activity
	Status	Short description of the status of the activity
	Result	Short description of the outcome of the activity
	Completed	Whether the activity has been completed
	Involvements	Set of actors involved in the activity and how each is involved
	Subactivities	Sequence of activities representing a breakdown of the work in this activity
	Events	Set of events related to the activity and how they are related
Actor	Resources	Set of resources related to the activity and how they are related
	Name	
Resource	E-mail	
	URI	Pointer to the actual resource
Event	Label	Description of the resource
	Start date/time	
	End date/time	
	Description	

dynamic Web page. Activity data can also be accessed through the Jena Java** API using constructs such as:

```
Resource focus=model.getResource(focusuri);
System.out.println("Title: " +
focus.getProperty(DC.title).getString());
```

which performs the same function as the JSP example. The other way to access activity data is by

using the Joseki HTTP API, which enables subsets of the RDF graph to be queried. The results are returned in an RDF serialization format such as RDF/XML.

For modifying RDF data, we provide an abstraction of a higher level than one which simply adds and removes statements from the model. The Unified Activity ontology specifies the data model but not the set of operations that can be performed on that data. Using the “raw” Java API to make changes can result in data graphs that are inconsistent with the activity ontology. To ensure consistency, we have developed an abstraction called a “command” that transforms the RDF data graph in a specific manner. A command, which is executed in a logged, reversible transaction, ensures that the resulting model continues to be consistent with relevant ontologies. While we have prototyped a scripting language encoded in RDF for developing compound commands, our deployed system uses plain Java code to implement commands. Examples of commands include creating a new activity, modifying a person’s involvement, and deleting an event.

The Jena Graph API provides access to one of the most compelling features of RDF: the ability to create a composite graph from different data sources, including virtual and logical ones. Because nodes and relationships in the RDF graph are identified by URIs, it is straightforward to build a composite graph by overlaying multiple graphs on top of each other. We have exploited this feature to build a framework for dynamically integrating external data sources into the core RDF model. For example, we have integrated our enterprise directory into the activity model, which means that activities can reference the name, e-mail address, or job title of anyone involved in the activity automatically, without having to explicitly add these properties to the model. This mechanism is what enables semantic-level integration, and theoretically enables any data source to be mapped into the Unified Activity schema while preserving its own semantics or leveraging other ontologies.

Queries of the composite graph are known as “federated queries” because they relate data from different data sources without moving all of that data into a central index. Our framework provides some support, particularly caching and prefetching, to improve the performance of these federated

queries, but does not resolve all the performance problems intrinsic to this configuration.

We have also built an access control model on top of our graph framework. This model works by post-filtering, based upon the identity of the user and the policies encoded in the RDF data, the results of data that is read. Users who do not have access to data simply do not see it. Likewise, calls to modify the repository can be checked at this level as well. As a post-filtering approach, the performance of this access control implementation is limited. In a sample worst case scenario, someone may search for all activities containing the term “the” in a system with 100,000 activities; 50,000 activities may contain that term, but the user may only have access to 2 of them. The system will retrieve 50,000 activities and filter out 49,998 of them from the display, resulting in potentially poor performance.

Discussion

We consider the role of RDF in activities an open question. Beyond the superficial (but real) costs of its unfamiliar representations and API, as well as performance issues, we have found the lack of support for complex relationships to be the largest problem with RDF as the data model for activities. RDF encodes binary assertions like `activity1 hasCreator person1`, which can be either present or not. It does not have a simple way to express an assertion like “activity1 involves person1 as of January 15, 2006 according to person2 and with involvement level of ‘responsible’.”

To express information about the relationship between “activity1” and “person1”, RDF relies on using three approaches, each having some variations. The first approach uses what we call a “relationship node” in which an intermediate node is interjected to express this relationship. All of the extra information about the relationship can be attached to the middle node “activity1-person1-relationship”, which is between the nodes “activity1” and “person1”. This relationship node can be either a blank node (meaning it has no identifying URI) or a regular URI node. If it is blank, then it must be referred to by the pair of “activity1” and “person1”. Moreover, queries and retrievals require traversing this extra node, which complicates matters for developers and impacts performance.

The second approach is known as “reification”. This is similar to the previous approach in that another

node is introduced to store the extra relationship information, but differs in that the extra node in this case is peripheral and the data can be accessed without knowledge about its existence. There are several ways to implement reification with different implications for the API and performance. We have observed that these implementations can become complex and can significantly hinder performance.

The third approach is to use multiple relationships to link together the same two nodes. For example, one might have the assertions `activity1 involves person1` and `activity1 hasResponsible person1`. This approach can be used to encode ordering by including in the model statements such as: `activity1 hasInvolvementOrdering seq1`, followed by `seq1_1 person1`, `seq1_2 person2`, etc. The disadvantage of this approach is that the multiple related assertions must be kept consistent, and having many paths between two objects seems to violate the principle of keeping a data model simple.

Rather than select a single approach, we use all three of these approaches in our current ontology. This state is confusing for developers. Worse, however, is the fact that we intend the RDF representation to be the logical and complete representation for activities. As such, we cannot conceal these approaches, but rather must expose them in the API, query language, and activity representation.

ACTIVITIES AS A WEB SERVICE

In order to realize our vision of having activities integrate multiple applications, we have provided a lightweight REST²⁵ interface to activity data. We have designed two levels of REST APIs to interact with the server.²⁶ The lower-level interface (known as the “RDF-level API”) operates directly on RDF data structures. It is not activity-specific and gives complete freedom to clients. The higher-level interface (known as the “activity-specific API”) provides simplified access for clients who simply want to perform standard operations on activities.

While RDF provides a compelling set of features, the activity-specific API has enabled us to explore different ways of making activity data accessible in the context of other applications such as e-mail clients or Web browsers. In fact, we have found that there is often a decision to be made when integrating activities and other applications: namely, whether

(1) the application data should be mapped into the activity model or (2) the activity service should expose its data to a plug-in or extension for the other application.

For example, in the past we have chosen the first of these options when integrating e-mail into activities by mapping e-mail into a messaging ontology and implementing a limited e-mail client as part of an activity-management user interface (UI). Later in this paper we will present an alternative approach that extends an open-source mail application with activity data accessed through the Wax activity service. Both approaches allow users to see activities and related messages alongside each other. The advantage of the former approach is the uniform data model; the advantage of the latter approach is the reuse of an existing e-mail client application. As e-mail clients are complex and heavily used and have idiosyncrasies that users come to rely on, the latter approach is much more effective in this case. Moreover, when bringing activity data to client plug-ins, we have also found that an XML-based syntax appears easier and more familiar for plug-in developers than RDF, and that the need for flexibility in this context is limited.

By providing both RDF and activity-specific interfaces, we can continue to explore the flexible data model of RDF, while hiding the decisions we make about representation from the particular XML serializations that we expose to client applications. We anticipate, however, that much of the innovation with activity integration will be in the form of bringing activity data into the context of existing tools and applications, and that the activity-specific API will be the most widely used for this purpose. All of the client plug-ins we present later make use of the activity-specific API.

RDF-level interface

The RDF-level interface provides access to the RDF data model directly. The interface consists of three methods. The first method enables the querying of the RDF database by using the standard RDQL or SPARQL protocols. We have currently implemented the RDQL protocol and plan to implement the SPARQL protocol in the near future.

The second method enables modification of the RDF database by applying a “delta”, that is, a list of RDF-triples (i.e. basic subject-predicate-object state-

Table 2 Unified Activity API calls in the Wax system

Create a new activity
Delete an activity
Edit an activity (change its title or description)
Set the shortname
Add/remove a note/attachment/resource from the activity
Mark an activity as completed
Add/remove/edit a person in the activity with a particular type of involvement
Add/download an artifact (which is then uploaded and stored on the server)
Add/remove/edit an RSS or Atom feed
Add a new event (calendar entry with start/end times and a description)
Add/remove/edit a subactivity in an activity
Retrieve the activity specified by the URI or shortname
Retrieve all activities involving a particular user
Retrieve all people involved with a particular activity and all of its subactivities

ments) to be added and another list of triples to be removed. It is necessary to support additions and removals within a single transaction to ensure that the database is always in a consistent state.

The third method allows a client to register with the server in order to be notified when a change occurs in the database. We have currently implemented a simple triple-pattern-based notification mechanism, but we are planning to extend this by using a SPARQL query to specify the triples to be observed. A notification is sent if the result of the query changes in response to a database change; the notification contains the change in query results due to the database update.

One problem with the RDF-level interface is that it does not use commands to encapsulate changes made to the RDF data graph, and thus allows clients to make changes that leave the graph in an inconsistent state. Were this API to specify the commands available, however, clients would be restricted to using the functionality already encoded

on the server. We continue to investigate trade-offs between flexibility and reliability, such as this one.

Activity-level interface

In addition to the RDF-level interface, we also provide a higher-level interface that is activity-specific. This API provides three categories of functionality: managing activities, managing properties of activities, and searching for activities that match various parameters.

Each activity has a unique identifier which globally identifies it. This enables multiple activities to have the same title and not conflict with each other. However, because activity URIs are machine-generated and difficult to remember, we have also provided the ability to associate a human-specified “shortname” for each activity, which can be used in place of the full URI.

The API calls at the activity level are shown in **Table 2**. The API provides the ability to create, delete, and edit activities and to set and retrieve various activity properties and related entities, such as resources, feeds, events, and subactivities. The API also allows activities to be retrieved in various ways, such as by URI, by the people involved, and by related activities.

The retrieval queries currently return XML describing the activity by using a custom schema which we designed. The Atom standard²⁷ defines an XML format for data interchange that captures many of the important features of activities. In future work, we are planning to investigate the use of Atom for representing activity data, in addition to participating in standards work that defines a common interchange format for activities in order to enable multiple activity systems to interoperate.

THE WAX ACTIVITY MANAGEMENT SYSTEM

While we expect that much of the interaction with activities will take place with existing tools using integration points enabled by the activity service, we have found it useful to create a complete, Web-based user interface. The Wax Web UI provides access to all activity functions as a collaborative Web application. With this interface, users can add people to an activity, send e-mail to people involved in the activity, add related resources (Web pages, documents, RSS feeds, and other artifacts), schedule events and deadlines, and define subactivities.

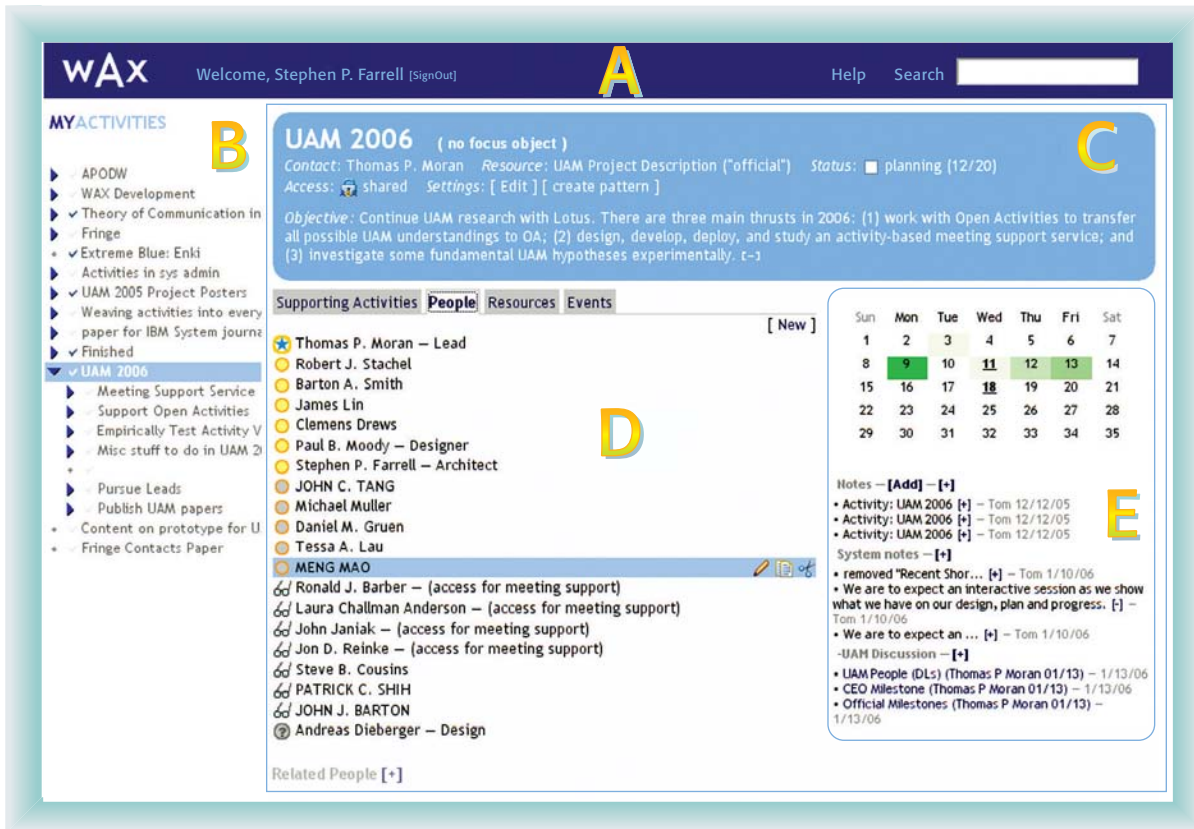


Figure 1
Wax system Web application

User interface

The Wax Web application shown in *Figure 1* provides a view of a single activity (the “focus”) and the user’s personal context. The UI is divided into five panels. The toolbar panel (A) at the top provides context-independent functions such as search and help. The “My Activities” panel (B) at the left is a hierarchical list of the user’s personal activities. Activities in this list can be opened dynamically, enabling the user to rapidly explore them. The center panel (C) displays basic information about the specific activity that is in focus, including the title, status text and check box, primary person, access policy, and description. It also provides interactive elements to change the status and to retrieve a form to edit other fields. The details panel (D) provides detailed information about people, documents, events, and other activities that are related to the focus activity and are displayed as tabs. The temporal panel (E) shows temporally organized information related to the activity, including mailing lists and RSS feeds, a

system-generated log of actions, and a calendar that highlights the days on which the most activity has occurred.

Example

As an example, a user named Sally may want to create an activity to plan for the “Acme Demo”. This demo takes place on a particular date. There are slides and applications to organize and prepare, and there are other people who will be helping out. Sally begins by selecting the “Acme” activity and creating this demo as a subactivity of it, specifying the title, description, access rights, and other properties of the new subactivity. To add more detail to the activity, Sally clicks on it, which makes it the focus activity. The details and temporal panels (D and E) are blank, so Sally sets about populating them. She switches to the “People” tab and clicks on “[New]” to add the other members, looking up her colleagues in the corporate directory from the dialog that appears, describing the role the person is playing in the activity and his or her involvement in more

detail, and indicating whether the person should be notified by e-mail of his or her involvement in the activity. She then clicks on “Supporting Activities” and creates new subactivities (sub-subactivities of the “Acme” activity) for the various steps in preparing for the demo.

Using similar steps, Sally associates with the subactivity various resources such as Wiki pages or mailing lists, deadlines, and milestones. As she adds these resources and objects, some information is automatically populated by hard-coded auto-discovery heuristics. For example, after adding a Wiki page, our heuristics automatically discover the change log for the page, and recent changes are added to the temporal panel (E). Similarly, recent mailing-list entries are discovered. The auto-discovery heuristics identify contributors to the Wiki page who are not already listed as involved in the activity, and display them at the bottom of the “People” tab as possibly related people.

Sally’s colleagues will receive e-mail saying that they have been added to the activity. When they visit the link in this e-mail, they will see an inline dialog box noting that they are part of this activity and offering to add it to their own My Activities panel. Because it is a shared activity, they all have permission to make changes including adding other members, resources, and deadlines. All changes are logged and appear in the temporal panel.

After this activity is created, all of the members can access it through the Wax Web interface and also through other components, thus integrating it into the tools they use routinely including e-mail, calendar tools, Web browsers, and productivity applications.

By creating this activity in Wax, Sally has not only populated the Wax Web application; she has also placed this activity in the context of the work practices of her colleagues through these integration components. The next section describes how activities can be interwoven into other tools using the Wax APIs.

CLIENT INTEGRATION AND USER INTERFACES

Several applications and extensions to third-party applications have been built with the activity-level

interface, including those related to e-mail tools, Web browsers and desktop search tools.

E-mail integration

An e-mail client can be modified to display an activities pane. Each activity in this display can contain a set of e-mail messages (notes) and the set of people involved in the activity. Although only notes are currently supported, we plan to extend the plug-in to support the other types of resources that can be associated with activities, such as attachments, RSS feeds, and subactivity information.

The Activities pane of the e-mail client shows all the activities that involve both the current user and the sender of the currently selected e-mail message. The activity list is prioritized by relevance, using the SimOverlap metric described in Reference 28; activities whose members are most similar to the set of recipients in the message are displayed at top of the list.

A new activity can be created by right-clicking on an e-mail message. Activities created in this way are automatically created on the Wax server as well. One of the primary benefits of using this interface to create activities is that the set of involved people is automatically determined from the e-mail. New messages can be associated with an activity by dragging and dropping them onto the Activities pane, and they are added to the activity as notes.

One of the primary benefits of this collaborative approach to managing activities is increasing activity awareness. Every participant in an activity sees the most up-to-date relevant activities displayed in his or her activity pane. For example, if Sally creates a new activity involving her and Bob, the next time Bob receives an e-mail from her, he will see the new activity displayed in the contextual activity pane, thus making him aware that Sally created the activity in which he is now involved.

Another benefit is collaborative activity management. Shared activities are a collaborative artifact: any changes to an activity are made visible to all participants of the activity. Thus all participants share responsibility for updating the activity, and all members benefit from the organizational work of other members. If Sally adds a particular message to

an activity because it is relevant, Bob benefits from her organizational work.

Web browser integration

The Wax Web site provides RSS feeds for activities. Using these feeds and Mozilla's live bookmarks feature, users can get one-click access to their activities directly from the Web browser.

We also provide an activity bar plug-in for Mozilla Firefox** browser that shows all activities which list the Web page currently being viewed as a resource. For example, a Web page can be added as a resource for the Wax Development activity. Users visiting this Web page would see the Wax Development activity listed in the activity bar as a relevant activity. In addition, users can associate a Web page with any of their activities by navigating to it in a browser and pressing the "Add page to activity" button. This functionality is similar to that provided by social bookmarking services such as the "del.icio.us" Web site,²⁹ where, instead of tagging a document, the user is declaring a more formal relationship between the current Web page and an activity.

The activity bar also is integrated with our online enterprise directory. When a user's profile page is visited, the activity bar displays all activities in which that user is a participant. The button on the right also changes to "Add user to activity." This capability uses pages in the enterprise directory as proxies for people in the organization, and makes it easy to add people to activities by simply viewing their page in the directory.

We have also considered the possibility of creating a new activity from a Web page. The title and summary of the Web page could be extracted as a description of the activity, for example, in the same way that the e-mail integration application extracts them from an e-mail message. The list of people involved in the activity could be extracted from the people mentioned on the Web page, and a description of the steps involved in the activity could be mined from the text of the Web page.

Google desktop integration

We also provide a client plug-in to integrate our activity system with the desktop search product by Google. This plug-in adds a content box to the Google desktop display that shows the list of one's

activities. Each item shows the title of the activity and a check box with the completed status of the activity. Clicking on an item brings up a popup window showing the objective of the activity. Double-clicking on an item brings up a Web browser showing the Wax page for that activity.

Other applications

Activities can be integrated with other applications such as word processors and spreadsheets by using a similar approach. With these tools, users should be able to save and store files directly as resources of the activities in which they are involved, and they can be made aware of activities that are related to the document that they are currently viewing.

Because involvements are an integral part of activities, any display of relevant activities can also include a list of relevant people. The users should be able to start an instant chat, for example, directly from a related-activities display. The transcript from that chat could optionally be saved as a resource of the activity.

In similar ways, activities can provide context for calendar use, project management, and video conferencing. Other domain-specific tools can provide even richer levels of integration. For example, a software development environment might associate a source code repository with an activity, with packages within that repository treated as subactivities and logs treated as resources. The activity could serve as an integration point between the development environment, source code management, requirements documents, discussion tools, and team members. Moreover, it can link to office tools, enabling a person viewing the requirements document, for example, to navigate to the source code, developers, and defect reports through the activity. The goal is for the activity service to support activities as an essential concept linking related information, tools, and resources in the context of the tools people already use.

Discussion

While they have not yet been widely deployed, our experience with these client plug-ins has led us to conclude that integrating activity data into existing applications has several benefits. Displaying activity data within applications that people already use (such as e-mail and Web browsers) enhances activity awareness. Adding resources and people to

activities is significantly facilitated, as is the creation of new activities that reuse existing context (such as the currently viewed e-mail message or Web page). These features make it more likely that people will use the activity system.

EXPERIENCE AND TESTING

We built the Wax prototype service to explore a unified activity representation and an integration architecture to support activity management. Our exploration of integrating Wax with other systems was presented in the previous section. We were not trying to build a robust system which was fully integrated into IBM's enterprise applications (such as Lotus Notes); rather we gave the Wax system enough functionality so that we could try it ourselves, open it to colleagues in related projects, and support targeted activities by selected non-technical groups.

The Wax system was adequate for this level of testing. The functionality, described in a previous section, supported basic collaborative activity representation and management. The user interface employed AJAX (Asynchronous JavaScript** and XML) techniques (e.g., reorganizing subactivity order by "drag-and-drop") to make its use easy and intuitive. The response-time performance was satisfactory, and continual debugging and enhancements kept the system at a usable and useful level. Finally, we were quick to restart the system when it crashed, and we frequently backed up the data users entered.

The Wax system was in use for 10 months, starting in 2005. Over 2000 activities were created by almost 200 different users. Most users were simply trying out the system, but many were doing real work with it (roughly 40 users who had more than 15 activities on their My Activities lists.) In addition, seven users explored using the Wax system to organize their personal activities. It is interesting that all but one of these users included collaborative activities. Thus the organization of the activities was personal, but the activities organized were shared. Other kinds of activities were also explored: planning (3), presentation (14), small projects (12), and trips (3).

Our focus in testing Wax was to support the six major activities listed in *Table 3*. The first three were activities related to the UAM project. We initially used the Wax system to keep track of

defects and feature requests; Wax proved somewhat useful for organizing these items into subactivity structures. The defect tracking work was carried out in SourceForge,³⁰ which is optimized to support a commonly accepted schema for defect tracking that is integrated with our code development in Eclipse**. But Wax was still somewhat useful to some users as a means for displaying an overview of the main features as well as aggregated feeds from the developers' mailing lists and from the UAM content database. Although there were only 10 people on the UAM project, we included several people outside our project as observers of the activity, because the activity structure itself served as a way to communicate the range of our work with them.

Case study: Organizing a complex event

We tested the Wax system by using it to manage a significant activity; namely, organizing the Almaden Research Center site-wide celebration event for IBM Research's 60th anniversary. This activity involved a committee of 16–18 people, each with different responsibilities (e.g., hiring caterers, rearranging the cafeteria layout, inviting guest speakers). Eighteen subactivities were created, mostly by the person in charge of the event. However, of those 18, two subactivities were created by a second user. Most of the users modified one or more of the subactivities during the course of the activity, by updating its status or adding resources.

The Wax system was used as the primary organizational system for coordinating this event. The group held weekly meetings, during which the activity display in the Wax system was projected onto a screen in order to see what needed to be done and track the group's progress. Before each meeting, the organizer would send e-mail to the person responsible for each subactivity, asking him or her to update the status of the activity in the Wax system. This practice made it unnecessary for the committee members to prepare slides for the meeting. During the meeting, additional information that emerged was added directly to the activities so that by the end of the meeting, everyone was able to see the action items for which they were responsible.

A survey was distributed to the activity participants afterwards in order to gauge their reactions to the system. Users agreed that the system was useful. They believed the primary value was that it

provided awareness of status and asynchronous communication. The committee consisted of members who did not normally work together; the activity provided a way for these diverse members to communicate with each other effectively, particularly by providing asynchronous communication. For instance, someone working over the weekend could check the status of an activity to see whether it had been completed, without needing to e-mail or call the person responsible for the activity. The structured nature of activities also helped to organize this status information in a way that made it easier to find information, as compared with searching through e-mail in order to find the most recent update. Another useful aspect of the structured representation was the ability to express responsibility by placing a name next to each part of the activity. In addition, because much of the coordination work of the activity could be done using the Wax system, the group found that it was not necessary to meet every week to maintain progress.

One observation was that the field “description” in the activity was overloaded to contain current information about the status of the activity. (The description field has since been renamed “objective”, in part due to this experience.) Our original intent for the description field was to contain a description of the goal or purpose of the activity. However, we observed that users often placed an ongoing status update of the work product into this field, such as the master agenda for the event, or the fact that a guest speaker had accepted an invitation to participate. We believe this was because the description field is prominently displayed at the top of the activity page, so it provides a convenient place for users to store information that they wish to make immediately accessible to anyone looking at the activity. In contrast, placing this information in the “Resources” section of the activity would mean having to click through a few screens before this information could be viewed.

A further benefit to using the activity system was in transferring knowledge from one person to another. In the middle of the planning process, a committee member was replaced. It was possible to make the new member current because all the pertinent information was recorded in the activity. If this information had been disseminated via e-mail, it would have been much more difficult to find all the

Table 3 Major activities supported by the Wax system

Activity	Number of People	Number of Subactivities
Wax development (defects and features)	21	31
UAM year-end 2005 presentations	20	27
UAM project planning for 2006	60	100
Organizing a site-wide anniversary event	23	20
Organizing the NPUC (New Paradigms in User Computing) 2006 Workshop	7	22
Managing accommodation of the Almaden student interns for 2006	111	918

relevant information and forward it to the new committee member.

The study also revealed some limitations that need to be addressed in a future version of the system. One common complaint was that the activity system needs to have better integration with formal business processes and external applications. For example, the process for reimbursing a guest speaker’s travel expenses involves filling out a form with the speaker’s name and affiliation. Because this information is already entered into the activity, it would be helpful if the reimbursement process could retrieve it from the activity so that it does not have to be entered twice.

Another common request was the ability to generate reports or summaries based on the information entered into the activity system. For example, one use would be to determine how much of the total budget has been spent by extracting this information from the subactivities. As another example, if the Wax system had been used to organize a weekly seminar, it would be useful to be able to print out a summary of all the talks that had been scheduled over the course of the past year. While generating these reports is currently possible by navigating to different screens in the system and copying and pasting data into a new document, one of our goals for future work is to explore mechanisms for

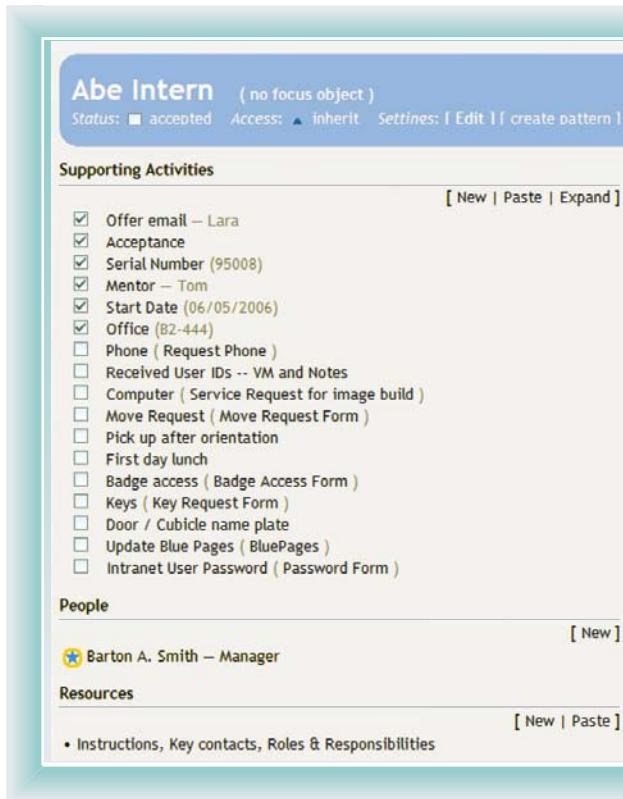


Figure 2
Example of Wax system view of intern accommodation activity pattern

enabling end users to do customized report generation.

Case study: Managing a business process

We explored how well the Wax system could support a common business process by using it to coordinate the process of setting up accommodations for 85 new summer interns at Almaden. The existing process consisted of a list of steps that was e-mailed to the manager of each intern, detailing what needed to be completed before and shortly after the arrival of each intern. These included assigning a mentor, assigning office space, requesting a telephone and voice mail account, arranging for a computer and making it ready, verifying creation of computer accounts, requesting badge access to appropriate rooms, and requesting keys. Some of these items required the use of formal corporate business processes or applications (e.g., badge access), and some required coordination among two or more people (e.g., assignment of office space).

While the preceding list was common to the Research Division, the Almaden laboratory needed to have a local process that not only conformed to the corporate requirements but also supported the actual practices at the Almaden site. This situation provided a good test of both the activity pattern concept and the information-sharing capabilities of the Wax system.

The activity pattern for managing intern accommodation was created collaboratively. We met with the Human Resources (HR) department to request their cooperation and understand their requirements. After this, we interviewed a group of managers and department administrative assistants (AAs) and technical assistants (TAs) who would be involved in performing some of the steps. On the basis of this input from potential users, we created an intern-accommodation activity pattern, which included 17 subactivities, one for each of the steps. The goal of the design was to provide, in a single view, the checklist of items to be done, the status of each item, and a resource for doing it if applicable. The AAs and TAs also voiced the need for a summary view so that they could check the status of certain items, for example, office assignments, for the list of interns in their departments. We built a special view that pulled information from the individual activities for each intern and presented it in a table (the intern summary table).

A hierarchy of activities was built to organize the activities for individual interns by department. HR personnel were involved at the top level, having access to all entries. Managers, TAs, and AAs in each department were involved in the subactivity for their area, because users had told us that they wanted to see a list of only the interns relevant to them when they opened their view of the activity. As a default, each subactivity inherited the involvement of people from its parent, providing the appropriate access control at each level.

Figure 2 shows the Wax system view of one instance of the intern accommodation activity. The name of the intern is used as the title of the activity, and each of the supporting activities is one of the steps in the process. For each item in the supporting activities list, one or more of the following types of information from the supporting activity is shown: the status of the check box, the title of the step, the name of the person responsible for the step, a text

Name	Manager	Mentor	Start Date	Office #	Phone #	s/n	email	computer allocated
Computer Science Interns								
Anthony Interns								
A. Smith	Barton A. Smith	Tessa A. Law	6/19/2006	B2-423 I		4D4918		✓ T30 99 5AZ9X
B. Smith	Theo Public	Dr. Zee	6/5/2006			4D5940		
C. Smith	Barton A. Smith	Thomas P. Moran	06/05/2006	B2-423 Q		4D6528	xyz@us.ibm.com	✓ T30 99-5BA1P
D. Smith	Barton A. Smith	Allen Cperson	05/30/2006	B2-423 P		4D6236	xyz@us.ibm.com	✓ T30 99-5AL1L
E. Smith	Seldom Nutty	Heather Citizen	06/19/06			4D6203		
F. Smith	Jack B. Nimble		5/15/06	B2-433		3D0933		✓ T30 KP-96BG7
A. Smith	Theo Public	Theo Public						
H. Smith	Steven R. Public		5/11/2006			953207		
I. Smith	Theo Public	Thomas A. Edison	May 15, 2006			4D7052		
J. Smith	Jack B. Nimble	Sophia T. Oyster						
K. Smith	Jack B. Nimble	Sophia T. Oyster						
Buffalo Bill Interns								
L. Smith	China Ho	Lucy Public	6/19/2006	B1-255 (#6)		3D0952		✓ T30 78-ANBZ5
M. Smith	Shivman Virtual	Raja Publicman	5/15/2006	B2-237 (#A)		3D1763		
N. Smith	Shivman Virtual	Shivman Virtual	6/12/2006	B2-249		4D5701		
O. Smith	China Ho	Mario Public	5/15/2006	B1-255 (#7)		781953		✓ Intelli M Pro 78-G3245 clio0301.almaden.ibm.com
P. Smith	China Ho	When-Saran Li	05/08	B1-431	457-1146	4D6430	xyz@us.ibm.com	✓ ThinkCenter KC-GX4C2 tphan1.almaden.ibm.com
Q. Smith	China Ho	Tanya Citizen	05/22	B2-249		4D6431		✓ Intelli M Pro 78-G3145
R. Smith	China Ho	When-Saran Li	05/08	B1-431	457-1002	4D6400	xyz@us.ibm.com	✓ ThinkCenter KC-GG1D9 dpa.almaden.ibm.com
S. Smith	China Ho	China Ho	05/15	B1-419	7-1734	4D6564		✓ ThinkCenter KC-FW0XP clio0601.almaden.ibm.com
T. Smith	China Ho	China Ho	05/15	B1-419		4D6916		✓ ThinkCenter KC-FW0ZZ clio0602.almaden.ibm.com
U. Smith	China Ho	Tanya Citizen	05/22	B2-249				✓ Intelli M Pro 78-G3276
V. Smith	Huey Long	Stephen Foster		B1-255 (#4)				✓ ThinkCenter KL-HAF34
Grandyman Interns								
W. Smith	Tyrone W. Grandyman	Alexa Efram	05/22/06			4D6208		
X. Smith	Catharine Hopefull							
Y. Smith	Catharine Hopefull							

Figure 3
Example of intern summary table

string describing the status or result of the step, and a link to the resource, such as an online form, used for accomplishing the task. For example, the badge access form is the corporate application used to request access to a specific laboratory.

In the “people” section of the onboarding checklist, the person who was to be the intern’s manager was listed as responsible. The “resources” section contained a link to a document with instructions on how to use the Wax system to coordinate the intern orientation process and a link to general help.

Figure 3 is an example of the intern summary table for one department. This table allowed the department TA to view the status of preparations for each intern in the department and to track the assignment of office space. It also allowed the person responsible for telephones to get the information needed for each intern (office number, e-mail address, and serial number) to set up the voice mail account and assign the phone number. In the table, the name of the intern is a link to the activity for that intern, into which new data can be entered. The name of each

manager and mentor provides a link to their entry in the corporate directory.

As of this writing, data for 54 interns and 38 mentors has been entered into the system by 22 managers and TAs. Our experience thus far has provided several insights. The use of an activity pattern, based on the actual practices of workers in addition to business process requirements, can be a valuable tool in coordinating repeated business activities. Summary views are required to meet the needs of users who play different roles in collaborative processes. The pattern-creation process should allow customization of page layout and organization, labels and headings, and task-specific instructions to make the transfer of the practices embodied in the pattern to new users easier.

SUMMARY AND FUTURE WORK

In this paper, we have presented an approach to collaborative activity management based on a shared, semantic representation of activity. We have described how activities can be represented using RDF and defined an ontology that captures the

common properties of activities. We have made the case for activities as a service, enabling other applications to make use of activity data and giving users the ability to manage their activities from within the context of the applications and tools they already use. To support this vision, we have developed the Wax platform that provides two lightweight Web APIs for accessing activity data. We have presented the Wax Web UI, which was built on top of the Wax platform, and several client plug-ins that enhance existing applications with activity data. Finally, we have evaluated the Wax system by using it to manage several large activities. Our experience with the system demonstrated that it had concrete benefits for the people collaborating on these activities, including better communication, easier knowledge transfer for people joining the project midstream, and a reduced need for in-person meetings.

We have only begun to explore the problems of supporting collaborative activity management, and there are many directions for future work. On an architectural level, we have not yet concluded that RDF is the best representation for capturing the flexible yet semantically coherent activity structures we want to support. Future work will explore alternatives to the RDF model and contrast them in terms of expressive power, ease of authoring activity applications, and robustness to changes in the schema. We are currently exploring the use of an activity vocabulary to organize work that can be managed in the form of Atom feeds. On a user experience level, we plan to conduct more user evaluations of the system in order to determine which aspects of an activity management system are most important to users and how we should grow the system to best meet users' needs. Finally, we are exploring how work supported by activity patterns can be more productive.

*Trademark, service mark, or registered trademark of International Business Machines Corporation.

**Trademark, service mark, or registered trademark of 3M Corporation, Sun Microsystems, Inc., Mozilla Foundation, or the Eclipse Foundation, Inc. in the United States, other countries, or both.

CITED REFERENCES

1. P. Moody, D. Gruen, M. J. Muller, J. Tang, and T. P. Moran, "Business Activity Patterns: A New Model for Collaborative Business Applications," *IBM Systems Journal* **45**, No. 4, 683–694 (2006, this issue).

2. S. Whittaker and C. Sidner, "Email overload: Exploring Personal Information Management of email," *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '96): Common Ground*, ACM Press, New York (1996) pp. 276–283.
3. C. Hill, R. Yates, C. Jones, and S. L. Kogan, "Beyond Predictable Workflows: Enhancing Productivity in Artful Business Processes," *IBM Systems Journal* **45**, No. 4, 663–682 (2006, this issue).
4. T. P. Moran, A. Cozzi, and S. P. Farrell, "Unified Activity Management: Supporting People in eBusiness," *Communications of the ACM* **48**, No. 12, special section on Semantic eBusiness Vision, 67–70 (December 2005).
5. T. P. Moran, "Unified Activity Management: Explicitly Representing Activity in Work-Support Systems." *Proceedings of the European Conference on Computer-Supported Cooperative Work (ECSCW 2005), Workshop on Activity: From Theoretical to a Computational Construct* (2005), <http://www.daimi.au.dk/~bardram/ecscw2005/papers/moran.pdf>.
6. T. P. Moran, "Activity: Analysis, Design, and Management," in *Theories and Practice in Interaction Design*, S. Bagnara and G. Crampton Smith, Editors, Erlbaum Press, Mahwah, NJ (2006).
7. M. Muller, W. Geyer, B. Brownholtz, E. Wilcox, and D. Millen, "One Hundred Days in an Activity-Centric Collaboration Environment Based on Shared Objects," *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2004)*, ACM Press, New York (2004), pp. 375–382.
8. Unified Activity Management, IBM Research, <http://www.research.ibm.com/uam>.
9. R. Medina-Mora, T. Winograd, R. Flores, and F. Flores, "The Action Workflow Approach to Workflow Management Technology," *Proceedings of the ACM Conference on Computer-Supported Cooperative Work (CSCW '92)*, ACM Press, New York (1992), pp. 281–288.
10. S. Dustdar, "Caramba—A Process-Aware Collaboration System Supporting Ad Hoc and Collaborative Processes in Virtual Teams," *Distributed and Parallel Databases* **15**, No. 1, 45–66 (January 2004), <http://dx.doi.org/10.1023/B:DAPD.0000009431.20250.56>.
11. T. Kreifelts, E. Hinrichs, and G. Woetzel, "Sharing To-Do Lists with a Distributed Task Manager," *Proceedings of the European Conference on Computer-Supported Cooperative Work (ECSCW '93)*, Kluwer Academic Publishers, Dordrecht, Netherlands (1993), pp. 31–46.
12. W. Wang and J. Haake, "Supporting User-Defined Activity Spaces," *Proceedings of the Eighth ACM Conference on Hypertext (HYPERTEXT '97)*, ACM Press, New York (1997), pp. 112–123.
13. Groove Virtual Office, Groove Networks, <http://www.groove.net/home/index.cfm>.
14. *Context and Consciousness: Activity Theory and Human-Computer Interaction*, B. Nardi, Editor, MIT Press, Cambridge, Massachusetts (1996).
15. W3C Semantic Web Activity, Worldwide Web Consortium, <http://www.w3.org/2001/sw/>.
16. Resource Description Framework, W3C Semantic Web Activity, Worldwide Web Consortium, <http://www.w3.org/RDF/>.
17. *RDQL—A Query Language for RDF*, Worldwide Web Consortium, <http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/>.

18. SPARQL Query Language for RDF, Worldwide Web Consortium, <http://www.w3.org/TR/rdf-sparql-query/>.
19. Web Ontology Language (OWL), Worldwide Web Consortium, <http://www.w3.org/2004/OWL/>.
20. Dublin Core Metadata Initiative, <http://dublincore.org/>.
21. FOAF Vocabulary Specification, <http://xmlns.com/foaf/0.1/>.
22. RDF Calendar Workspace, Worldwide Web Consortium, <http://www.w3.org/2002/12/cal/>.
23. B. L. Harrison, A. Cozzi, and T. P. Moran, "Roles and Relationships in a Unified Activity Management System," *Proceedings of the ACM Conference on Supporting Group Work (Group 2005)*, ACM Press, New York (2005), pp. 236–245.
24. Java ServerPages Technology, Sun Microsystems, <http://java.sun.com/products/jsp/>.
25. R. T. Fielding, *Architectural Styles and the Design of Network-Based Software Architectures*, Ph.D. dissertation, University of California, Irvine, CA (2000).
26. E. M. Maximilien, A. Cozzi, and T. P. Moran, "Semantic Web Services for Activity-Based Computing," *Proceedings of the Third International Conference on Service-Oriented Computing (ICSOC 2005)*, LNCS 3826, Springer-Verlag, Berlin (2005), pp. 558–563.
27. AtomEnabled/Developers, <http://www.atomenabled.org/developers/>.
28. M. Dredze, T. Lau, and N. Kushmerick, "Automatically Classifying Emails into Activities," *Proceedings of the ACM International Conference on Intelligent User Interfaces (IUI 2006)*, ACM Press, New York (2006), pp. 70–77.
29. del.icio.us, <http://del.icio.us/>.
30. SourceForge.net, <http://sourceforge.net>.

Accepted for publication May 22, 2006.

Published online October 24, 2006.

Alex Cozzi

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (cozzi@almaden.ibm.com). Dr. Cozzi is a research staff member at the IBM Almaden Research Center in San Jose, California. He received an M.S. degree in computer science from the University of Milan, Italy, and a Ph.D. degree from the University of Dortmund, Germany. He has worked on computer vision and social network analysis. His research interests include human-computer interaction and data mining.

Stephen Farrell

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (sfarrell@almaden.ibm.com). Mr. Farrell is a senior software engineer at the IBM Almaden Research Center in San Jose, California. He has a B.A. degree in the history and philosophy of science and an M.S. degree in computer science from the University of Chicago. Mr. Farrell joined IBM Research in 1999 to develop systems which enhance the ability of people to work or collaborate. Some of his research areas include personalization of the Web experience, information programming, person-centric data integration, and relationship-oriented computing. Some of his projects include Fringe, which gives a person-centered view of enterprise information, and Enki, which does the same for personal information sources. He is also lead architect of the Unified Activity Management project.

Tessa Lau

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (tessalau@us.ibm.com). Dr. Lau is a research staff member at the IBM Almaden Research Center. She completed her Ph.D. degree in computer science at the University of Washington in 2001. She is interested in information management, particularly personal information, and how people interact with and customize their working environment. She has done significant work in the area of programming by demonstration, giving end users the ability to automate repetitive tasks simply by showing the system how to perform the task a few times.

Barton A. Smith

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (basmith@almaden.ibm.com). Dr. Smith received a B.S. degree in chemistry from the University of Texas at Austin in 1972, and a Ph.D. degree in physical chemistry from Harvard University in 1977. After two years at Stanford University, where he was a National Science Foundation Post-Doctoral Fellow, he joined the IBM Research Division in San Jose, California, in 1979. In his 26 years as a research staff member in IBM Research, he has worked on fundamental problems in polymers, materials, and computer science, and on IBM products including magnetic disks, optical disks, magnetic tape, typewriter ribbon, circuit boards, integrated circuits, multichip ceramic circuit modules, optical data transmission, displays, and ThinkPads®. His most recent work centers on understanding how people use information technology and improving the human-computer interaction experience. Dr. Smith currently manages the Human Interface Research group in the Computer Science department at the Almaden Research Center.

Clemens Drews

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (cdrews@us.ibm.com). Mr. Drews is a software engineer and currently works on the Unified Activity Management project. He has a B.S. degree in technical computer science from the University of Applied Sciences in Hamburg, Germany. He joined IBM Research in 1998. During his career at IBM, Mr. Drews has worked on projects ranging from short-range wireless communication gadgets to large multiuser collaboration tools.

James Lin

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (jameslin@us.ibm.com). Dr. Lin is a research staff member at the USER (User Sciences and Experience Research) group at the IBM Almaden Research Center. His research interests include user interfaces for collaboration, end-user programming, and creating tools for designing next-generation user interfaces. He received a Ph.D. degree in computer science from the University of California at Berkeley.

Bob Stachel

IBM Software Group, 1 Rogers Street, Lotus Development, Cambridge, Massachusetts 02142 (bob_stachel@us.ibm.com). Mr. Stachel is a software developer in the Collaborative User Experience Research Group. Since joining IBM Research in 1998 as a senior software developer, he has worked on projects in support of activity-centric computing, lightweight collaboration, expertise location, and knowledge management. Mr. Stachel joined Lotus in 1985, and was an architect on development teams for the Domino™ Merchant (electronic commerce), Lotus Notes Newsstand (electronic publishing), and Lotus Improv (spreadsheet) products. He has a B.A. degree from Brandeis University in computer science.

Thomas P. Moran

IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, California 95120 (tpmoran@us.ibm.com). Dr. Moran is an IBM Distinguished Engineer and leads the Unified Activity Management project. He was one of the pioneers establishing the field of human-computer interaction (HCI) within computer science, co-authoring (with Allen Newell and Stuart Card) the seminal book *The Psychology of Human-Computer Interaction* (1983). He was at Xerox PARC for 27 years as Principal Scientist and manager of User Interface and the Collaborative Systems research and as the Director of Xerox EuroPARC in Cambridge, England. Dr. Moran is founder and editor of the journal *Human-Computer Interaction*. He is an ACM Fellow and recipient of ACM SIGCHI's (special interest group in human-computer interaction) Lifetime Achievement Award. ■