# A Unified Abstraction for Messaging on the Semantic Web

Dennis Quan
IBM Internet Technology
1 Rogers Street
Cambridge, MA 02142 USA
dennisq@us.ibm.com

Karun Bakshi
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139 USA
karunb@ai.mit.edu

David Karger
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139 USA
karger@theory.lcs.mit.edu

## ABSTRACT

Since its inception, the Internet has been a hotbed of several successful communications channels, starting off with e-mail, Internet Relay Chat and Usenet newsgroups and more recently adding Web annotation, instant messaging, and news feeds. However, these channels were developed fairly independently, and in many cases their respective functionalities have grown to overlap significantly. For instance, users of these systems have separate identifiers for e-mail, chat, and instant messaging, and clients for these systems all have their own implementations of threaded message views. We believe these problems stem from a lack of a common user interface and data model. In this paper we use basic concepts from the Semantic Web and RDF to unify and model these seemingly disparate messaging paradigms. We also demonstrate a generalized user interface for messaging that uses the data model we have developed. From this process we realize a number of synergies that result from the reduction of overlap and the finer-grained control users are given over message composition, transmission, storage and retrieval.

## Categories and Subject Descriptors

H.5.3 [**Information Interfaces and Presentation**]: Group and Organization Interfaces – *collaborative computing, computer-supported cooperative work*.

## General Terms

Management, Human Factors

## Keywords

RDF, Semantic Web, e-mail, instant messaging, IRC, newsgroups, annotation, web logs, blogs, news feeds, user interface, messaging, collaboration

## 1. MOTIVATION

One of the greatest accomplishments of the Internet has been enabling communication in various forms. Perhaps the most notable technology in this space is e-mail, but instant messaging (IM), Internet Relay Chat (IRC), and Usenet newsgroups have also found widespread use for related applications. The success of these communications protocols can be attributed not only to the degree to which they address shortcomings of their non-digital counterparts but also to their ubiquity and prominence as standards.

These systems have been developed somewhat independently over the past half century and continue to be extended with new functionality that addresses the broadening needs of their users. While the systems share some common notions, such as e-mail addresses and MIME headers, the amount of perhaps unnecessarily duplicated infrastructure is becoming increasingly evident. For example, all of these protocols have different authentication and identification systems. The most obvious indicator of this phenomenon is the overlap in functionality in the clients for these systems. E-mail clients, instant messengers, and IRC clients all have widgets for displaying lists of people and means for notifying senders of a recipient's absence. Newsgroup readers and e-mail clients both have threaded message views and different mechanisms for filtering out messages from specific people.

This line of reasoning may seem like an impetus for a shared component architecture, but it extends further as the uses for the different communications channels have started to overlap. For example, the same tasks can be and are accomplished via e-mail and IM, e.g. sending quick messages for short term coordination, reminding others of pending tasks, and asking questions. Furthermore, multiple modalities may be involved in the completion of a single task: to notify a friend that you are going to be out for the day, you may start off by attempting to send an instant message, but if he or she is not online, you will likely switch to an e-mail client. If a user has a technical problem with a program, he or she may have to send a helpdesk request through several different channels separately.

Such a trend is to be expected as these systems are all addressing different aspects of the same fundamental problems of interpersonal and group communication. While in the beginning, the different communication channels crystallized functionalities specific to key activities, users have grown more reliant on these systems and are now bumping against the limitations of the abstractions. As a result, the opportunity exists to take a "bigger picture" look at the situation and to recast the problem in terms of a broader messaging abstraction. Once the existing systems are unified under a common model, we can also enhance all forms of messaging by incorporating features that are currently present only for specific messaging paradigms.

## 2. EXISTING MESSAGING SYSTEMS

A significant amount of research has been done regarding the major Internet messaging systems. Previous work has cumulatively uncovered the core strengths and limitations of each system. We use these findings to take a big picture view of the problem at hand and realize that despite their ostensible differences, all communication systems can be placed in context and compared by considering a handful of useful partitioning criteria, as diagrammed in Table 1. Considering each criterion in turn, we show how communication media that are similar in some respect tend to have similar properties, preferred uses and problems. Next, we present examples of some communication media that have various combinations of these aspects and have gained widespread acceptance in order to understand what particular niche they serve. We then use these findings as a starting point for our integration work.

**Table 1: Existing messaging systems**

| | **Synchronous (generally not persistent)** | **Asynchronous (generally persistent)** |
|---|---|---|
| **Private** | Instant Messaging | Mail |
| **Public** | IRC | News, Annotation, News feeds |

## 2.1 Synchronicity and persistence

Synchronicity captures the essence of conversation timing and serves as a fundamental divider of different communication modes. In asynchronous communication, the sender does not wait for a response and conversations are generally carried out over longer periods of time, with each party having the luxury of formulating a well thought out response. On the other hand, users exchange information relatively rapidly in synchronous communication, where a reply can generally be expected within a reasonable time period to facilitate an active dialog.

Inherent in the idea of asynchronicity is the notion of automatic persistence, which is at once both its boon and its bane. Asynchronous messages such as e-mails tend to be longer and automatically persisted. Whereas long term persistence supports capturing knowledge for future reference, it also allows extraneous information to add "noise" to the information environment, making it difficult to obtain and attend to important information [11]. On the other hand, short and to the point messages comprise synchronous communication, with persistence being an explicitly specified option due to the way these systems have evolved. Finally, synchronous communication, being closer to face-to-face communication, tends to be more informal and places greater importance on social interaction cues, e.g. response times, awareness of presence, etc. [9].

The real-time nature of synchronous communication media poses a number of problems. Voida and Smith both present evidence for IM and group chat indicating that managing context in a single conversation is challenging, requiring close attention by its participants to the progress of multiple threads in the conversation, and would benefit much from design enhancements [9] [6]. Furthermore, it seems that design enhancements needed to improve identification of thought boundaries in order to identify a turn, and interjection in order to claim a turn are equally applicable to IM and chat systems [9]. Furthermore, in synchronous systems, there is no way to ensure that a response appears in the right context since the display is a temporal sequence rather than a topical hierarchy [6]. Also, as the number of speakers goes up, they become difficult to distinguish. Finally, synchronous communication lacks immediate feedback on the listening status of the participants. Smith et al. point at threaded chat systems that may resolve some of these problems in certain situations such as decision making by allowing the conversation to remain focused, simplifying turn-taking and allowing easy access to recent comments. Nevertheless, such systems fail when the participant may be interested in multiple threads, each one competing for attention in a different portion of the UI.

## 2.2 Public versus private

Communication paradigms may also be grouped based on whether they support public access and dissemination of information where the recipients are unknown *a priori*, or whether they are intended for private, communication where the participants are known and can be selected. As usual, the notion of public may be restricted by other means, e.g. all employees of a particular company.

## 2.3 Current messaging systems

In this section, we discuss several communication mechanisms that currently exist, their particular properties, usage niche and existing problems. Specifically, we consider email, instant messaging, news groups, IRC style chat, shared annotation and news feeds.

### *2.3.1 E-mail*

E-mail, serving as a generalized asynchronous communication mechanism for social interaction and work-related collaboration, is perhaps the most widely used mode of digital communication. Whittaker et al. report that e-mail has evolved from an asynchronous communication mode to a focal point for task management and information organization simply because it serves as a mechanism for assigning and tracking work, as well as a receptacle of various kinds of information [5]. This is primarily due to the e-mail inbox being capable of maintaining context for related messages, simplifying information availability by collocating it and serving as a constant reminder of items needing attention. Furthermore, it makes available a single convenient, accessible, long-term archiving mechanism allowing easy filing for items in the inbox, or letting the inbox itself be the archive.

### *2.3.2 Instant messaging*

Nardi et al. describe instant messaging as a synchronous communication mode between two people that facilitates almost instantaneous exchange of short messages resulting in a casual conversation atmosphere [7]. Although the individual messages themselves may be short, immediate and rarely persisted, instant messaging allows maintenance of longer term sessions that allow awareness of presence of other parties, thereby facilitating longer term context maintenance and allowing continuation of the conversation. Unlike e-mail, users do not consider an IM session as a heavyweight activity requiring a formal addressing process, greeting and common ground determination prior to information exchange. This view is supported in how it is used: short queries for informational or coordination purposes and social interaction with others—all tasks that do not require long or complicated messages.

Despite its widespread use, IM introduces problems unique to its domain. For example, a request to converse may come at an inopportune moment and be considered disruptive due to its interruptive notification system. Also, robust designs should support mechanisms for finer-grained control over availability indicators without forcing users to resort to manual management of availability status of multiple online identities for the same person [9].

### *2.3.3 Newsgroups*

Newsgroups comprise perhaps the largest online communities that have resulted from the proliferation of the Internet [8]. Whittaker's findings on group discussion seem equally applicable to newsgroups in general [11]. Although similar to e-mail in being a persistent means of asynchronous messaging, newsgroups differ in one very fundamental way. Unlike e-mail and IM which are "by invitation only" paradigms, newsgroups allow public access to and participation in ongoing conversations. An interesting aspect of this mode of communication is that the general interest of the participants is well known or easily inferable, and hence establishment of common ground for a dialogue is fairly easy. Remarkably, only a minority of newsgroup users contribute a large portion of the discussion while the majority of users are content to be passive observers. According to Whittaker, group discussions function both as active dialogue for exchange of information as well as repositories of immediate and reapplicable knowledge embedded in archives of past discussions that can be searched by newcomers [11] Discussions on newsgroups provide a means for their members not only for interactive question/answer and debate, but also as a means of broadcasting reference information of general interest.

Unfortunately, newsgroups are subject to significant levels of irrelevant postings such that confusion resulting from the "noise" seems to lead to multiple discussions on the same topic. Whittaker argues that newsgroups should support easy change of communication mode (e.g. instant messaging when answers to urgent queries are needed). As it is, newsgroup users resort to e-mail as a means of privately continuing a conversation that started as a public post.

### *2.3.4 IRC and group chat*

Group chat systems, like other communication technologies, have come to support both social interaction as well as work collaboration such as discussion and decision making and group memory [11]. Much like newsgroups, chat systems tend to be publicly accessible, but the conversations tend to be ephemeral; the conversation is not stored in an archive. The lack of persistence is generally a by-product of the near synchronous nature of the communication mode. The conversations proceed so fast that responses to one statement in a given topic are interleaved with new topics or completely different threads, yielding an unintelligible sequence of messages whose utility as a future knowledge repository is limited.

### *2.3.5 Shared annotations*

Although annotation is not normally considered a form of communication, when it is used in a shared context such as peer revision, annotations gain many of the characteristics of newsgroup postings. One can observe that the primary distinguishing characteristic of an annotation is the specification of which document serves as the annotation's topic. Furthermore, collaborative annotation systems permit replies to be posted to annotations, giving these systems a notion of threading similar to those found in e-mail and in newsgroups. Indeed, web-based annotation products such as Microsoft Office 2000 allow users to post documents online on websites and enable users to participate in threaded online discussions [12]. Also of interest are recently developed annotation systems that permit both metadata and textual messages to be specified [13] [14].

### *2.3.6 News feeds and web logs*

Another interesting arena for messaging exists in the distribution channels provided by online news feeds and web logs (also known as "blogs"). Unlike other forms of messaging, news feeds and web logs are usually unidirectional, streaming messages (i.e., news articles) to a large audience. However, as is the case in the physical world, news-style distribution does not preclude bidirectional dialog from occurring. The analog of "Letters to the Editor" is sometimes provided in news feeds if a return e-mail address is included. As perhaps one of the more nascent forms of communication discussed here, news feed clients are perhaps the most lacking in the basic functionality possessed by client software for the other protocols.

## 3. APPROACH

In developing a unified framework for messaging, our approach rests on preserving the existing communication capabilities supported by the Internet, simultaneously attempting to address many of the problems they engender in their respective domains, and capitalizing on the synergy that results from treating different communication modalities as a single messaging system. In implementing these goals, we develop a robust infrastructure resting on a well defined ontology for messaging using the Resource Description Framework (RDF), a Semantic Web technology for integrating disparate systems and data together [4]. The robust infrastructure in turn facilitates addressing many of the UI problems and overlaps that exist. RDF is a portable format for describing semantic networks or labeled directed graphs [3]. Furthermore, RDF is a flexible, "semistructured" data model in which one can model a variety of concepts, from annotations to news feeds.

To realize the RDF data model, we are building support for unified messaging into Haystack, an information management project at the Laboratory for Computer Science at MIT [1]. The goal of the Haystack project is to develop a tool that allows users to easily manage their documents, e-mail messages, appointments, tasks, and other information. Haystack uses RDF to describe the connections between different documents in a user's corpus as well as the metadata concerning each document. Haystack's user interface exposes general tools for navigating the various kinds of information found in the user's corpus. A screenshot of Haystack is shown in Figure 1.

In order to apply RDF to the problem at hand, we give ontological specifications of how to represent messages, conversations, and people using RDF Schema. These representations generalize the notions of sender, recipient and reply threads and form the basis of our messaging data model. This model allows us to aggregate arbitrary types of messages thereby supporting the types of medium interchanges people often make (e.g. switching from a public post to a private e-mail discussion) while at the same time capturing the entire conversation to maintain message context that is so crucial in activities such as task management [5] [7]. Furthermore, by casting messages and conversations into RDF, we also gain a persistent description to which we can add additional metadata that will improve searches and other information retrieval techniques. Hence, users will be better able to reduce "noise" and manage information overload by being willing to file information and not worrying about not being able to find it later. Finally, a unified messaging paradigm implies that all information is collocated and hence conveniently accessible, which is crucial from a usability perspective [5].

As we hinted at above, a number of the issues with messaging systems today are user interface problems. Our prototype user interface built on Haystack attempts to generalize and unify many of the underlying themes present separately in each of the existing systems. For example, one major difference lies in the levels of convenience: quickly participating in a discussion is much easier in instant messengers and IRC clients than in e-mail and news readers. Also, another problem with messaging today is that user interfaces are not equipped to handle the huge volumes of messages that are often encountered today, as mentioned earlier. By creating higher-level organizational concepts such as conversations, we are able to consolidate messages with similar topics together, reducing the clutter in users' inboxes, while giving users more intuitive ways to navigate through their message corpora.

## 4. UNIFYING THE DATA MODEL

In this section we discuss the various elements of our messaging ontology. Figure 2 depicts the different elements of this ontology and how they interrelate by means of an example.

### 4.1 Messaging drivers

Our ontology is designed specifically to work with existing messaging systems. As a result, the base of our messaging infrastructure consists of a series of mail drivers capable of sending and receiving messages over protocols such as POP3, SMTP, and Jabber. When a message is to be sent, the system must be able to determine which of the available messaging drivers is best suited to delivering the message given the circumstances. In our system, messaging drivers are described as having type `msg:MessageSendService`.

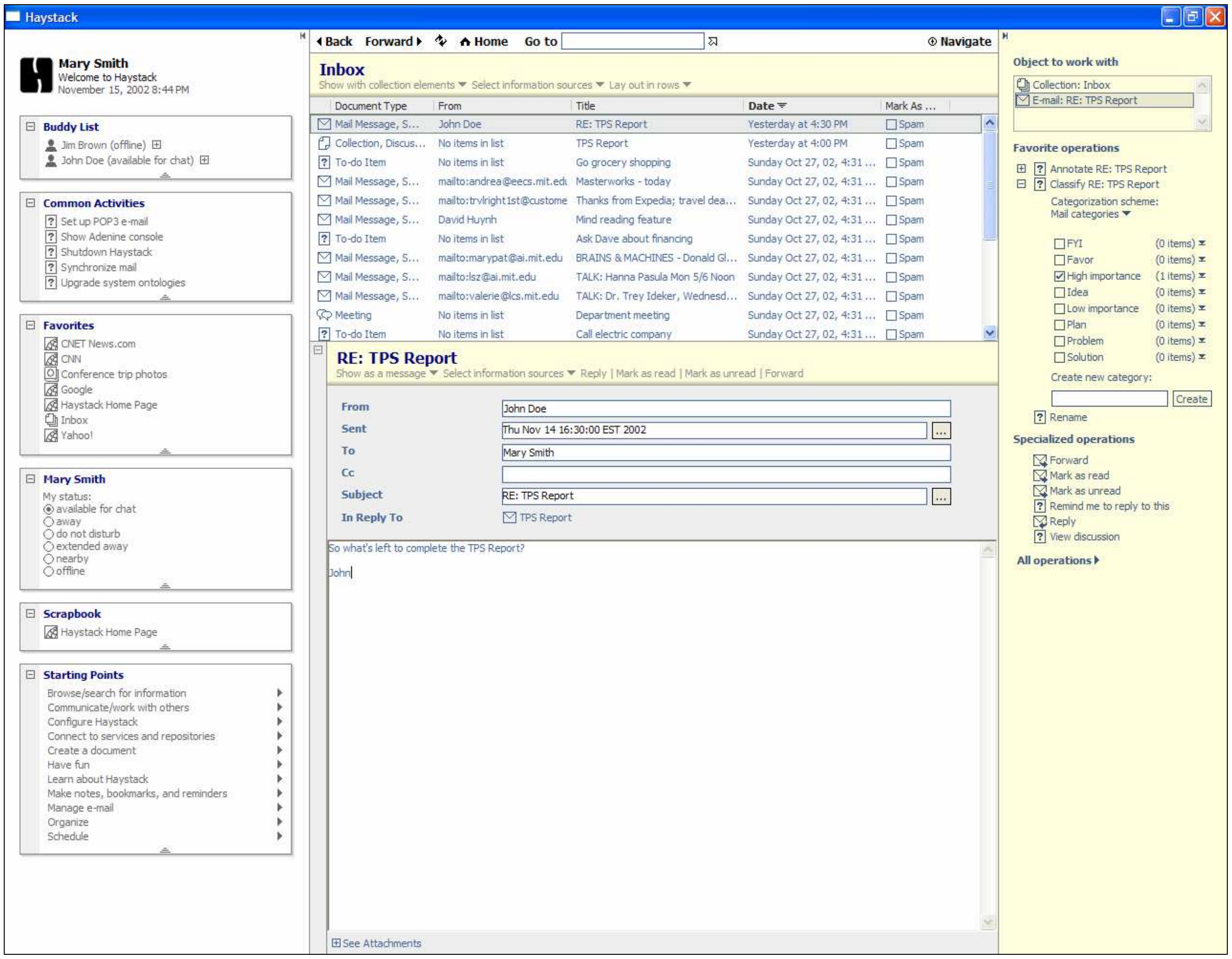
Messaging drivers are responsible for emulating functionality that is not normally available in the underlying protocol. For example, e-mail uses MIME headers to describe metadata concerning the messages, whereas IRC messages typically have no metadata. Techniques such as encoding messages in SOAP envelopes can be employed in these cases [16].

Messaging drivers are also responsible for incorporating messages into Haystack's RDF repository, which makes messages accessible via the user interface. This results in all messages being persistent.

### 4.2 Identity and addressing

As noted earlier, each messaging protocol currently maintains its own address scheme. For example, SMTP servers are programmed to route e-mail messages according to recipients' e-mail addresses. These addresses are represented directly in our ontology as URIs. In the case of e-mail, a well-defined scheme for forming a URI from an e-mail address is available, which simply directs the system to prepend the `mailto:` protocol scheme to the e-mail address. There have been similar schemes devised for the other protocols.

When a message is specifically directed to be routed by means of a particular address, the system needs to be able to resolve the address to a driver capable of interpreting it. We specify a base



| Document Type | From | Title | Date | Mark As ... |
|---|---|---|---|---|
| Mail Message, S... | John Doe | RE: TPS Report | Yesterday at 4:30 PM | ☐ Spam |
| Collection, Discus... | No items in list | TPS Report | Yesterday at 4:00 PM | ☐ Spam |
| To-do Item | No items in list | Go grocery shopping | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Mail Message, S... | mailto:andrea@eecs.mit.edu | Masterworks - today | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Mail Message, S... | mailto:trvlright1st@custome | Thanks from Expedia; travel dea... | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Mail Message, S... | David Huynh | Mind reading feature | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| To-do Item | No items in list | Ask Dave about financing | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Mail Message, S... | mailto:marypat@ai.mit.edu | BRAINS & MACHINES - Donald Gl... | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Mail Message, S... | mailto:lsz@ai.mit.edu | TALK: Hanna Pasula Mon 5/6 Noon | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Mail Message, S... | mailto:valerie@lcs.mit.edu | TALK: Dr. Trey Ideker, Wednesd... | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| Meeting | No items in list | Department meeting | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |
| To-do Item | No items in list | Call electric company | Sunday Oct 27, 02, 4:31 ... | ☐ Spam |

| | |
|---|---|
| From | John Doe |
| Sent | Thu Nov 14 16:30:00 EST 2002 |
| To | Mary Smith |
| Cc | |
| Subject | RE: TPS Report |
| In Reply To | TPS Report |

So what's left to complete the TPS Report?

John

| | |
|---|---|
| ☐ FYI | (0 items) |
| ☐ Favor | (0 items) |
| ☑ High importance | (1 items) |
| ☐ Idea | (0 items) |
| ☐ Low importance | (0 items) |
| ☐ Plan | (0 items) |
| ☐ Problem | (0 items) |
| ☐ Solution | (0 items) |

**Figure 1: Screenshot of Haystack**

class called `msg:Address` that represents addresses handled by mail drivers. This allows the system to recognize resources as being addresses without relying on the syntactic form of a resource's URI (e.g., whether the URI starts with "mailto:"). We can then derive classes such as `msg:EmailAddress` and stipulate that e-mail address resources be asserted to have this type.

Whereas most current messaging systems have a single identifier which is used for both identification and addressing, our unified messaging ontology necessarily distinguishes between the two concepts since the same person may have a different address for message delivery depending on the message type. However, it is not necessary for those sending messages to concern themselves with the specific address by which a message will be sent. Instead, people can be represented directly by means of the `hs:Person` class. Recipients and senders are specified by instances of the `msg:AddressSpecification` class. Address specifications can specify either a specific address or a person resource, or both. Addresses can be associated with people with the `msg:hasAddress` predicate.

Furthermore, some protocols support the notion of presence, allowing users to tell when their contacts are online and available for communication. We model this notion by associating `msg:Endpoint`'s with the `msg:onlineStatus` property. Drivers for protocols that support presence are responsible for keeping these properties up to date.

## 4.3 Messages

In order to incorporate the various forms of messaging available, we define the class `msg:Message` in a very general manner. In our system a message is a unit of expressive communication transported from one or more senders to one or more recipients. This definition allows us to unify the concepts of instant messages, e-mails, newsgroup postings, annotations, chat, and even articles delivered via news feeds.

However, care is needed to distinguish messaging from the general recording of information. We argue it is possible but not useful to define messaging to include all forms of information passing from one component to another. For example, does saving a document to disk constitute the sending of a "message"? Most users are unaware of the inner workings of a computer, and the transmission of a document from memory to disk occurs within the same "black box" as far as the user is concerned. Like all models, our ontology implicitly places a practical boundary around the types of objects we wish to describe. To achieve this, we wish to restrict our modeling of messages to include only communication where it is useful to acknowledge the transmission process from the sender's perspective. Messaging in the sense presented in this paper must therefore involve communication between either human parties or autonomous agents that are working as substitutes for human parties, such as forum moderators.

In addition, it is helpful to distinguish the idea of "audience" from "recipient". When a user posts a webpage to an Internet HTTP server, he or she is declaring the audience of that webpage to be the general public. However, the user is not "sending" this page to anyone; in other words, in the beginning the page has no "recipient". Contrast this situation with that of information being leaked to the press: a reporter may be the recipient of information, although the intended audience of that information certainly does

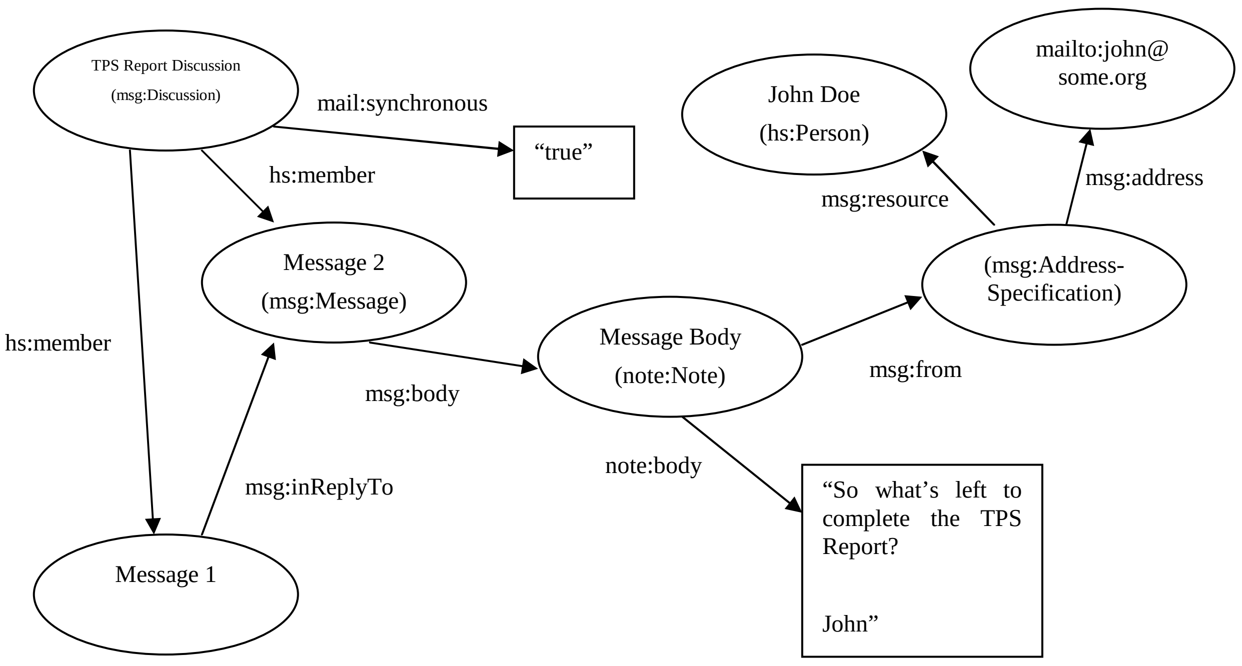


**Figure 2: Messages modeled according to our ontology (rdf:type's are indicated in parentheses; msg: prefix indicates http://haystack.lcs.mit.edu/schemata/mail# ontology; hs: prefix indicates http://haystack.lcs.mit.edu/schemata/haystack# ontology; and note: prefix indicates http://haystack.lcs.mit.edu/schemata/note# ontology for text documents)**

not include him or her. For our concept of messaging, we care only about message recipients, emphasizing the notion that messaging concerns only the aspect of transmission, not intended audience.

Messages come in various forms. The bulk of all messages are textual, but in the context of the Semantic Web it is also useful to provide for messages that conform to some established schema, such as meeting requests, money orders, or even bank statements. However, it is important to note that objects such as text documents, financial statements and currency can exist outside of messaging environments. The notion of message is independent and to some extent orthogonal to the notions of text documents and financial statements. Therefore, we define a message as an object for which a sender and a set of recipients are specified. A message also contains a body, which can be of any type known by the system.

## 4.4 Threads and conversations

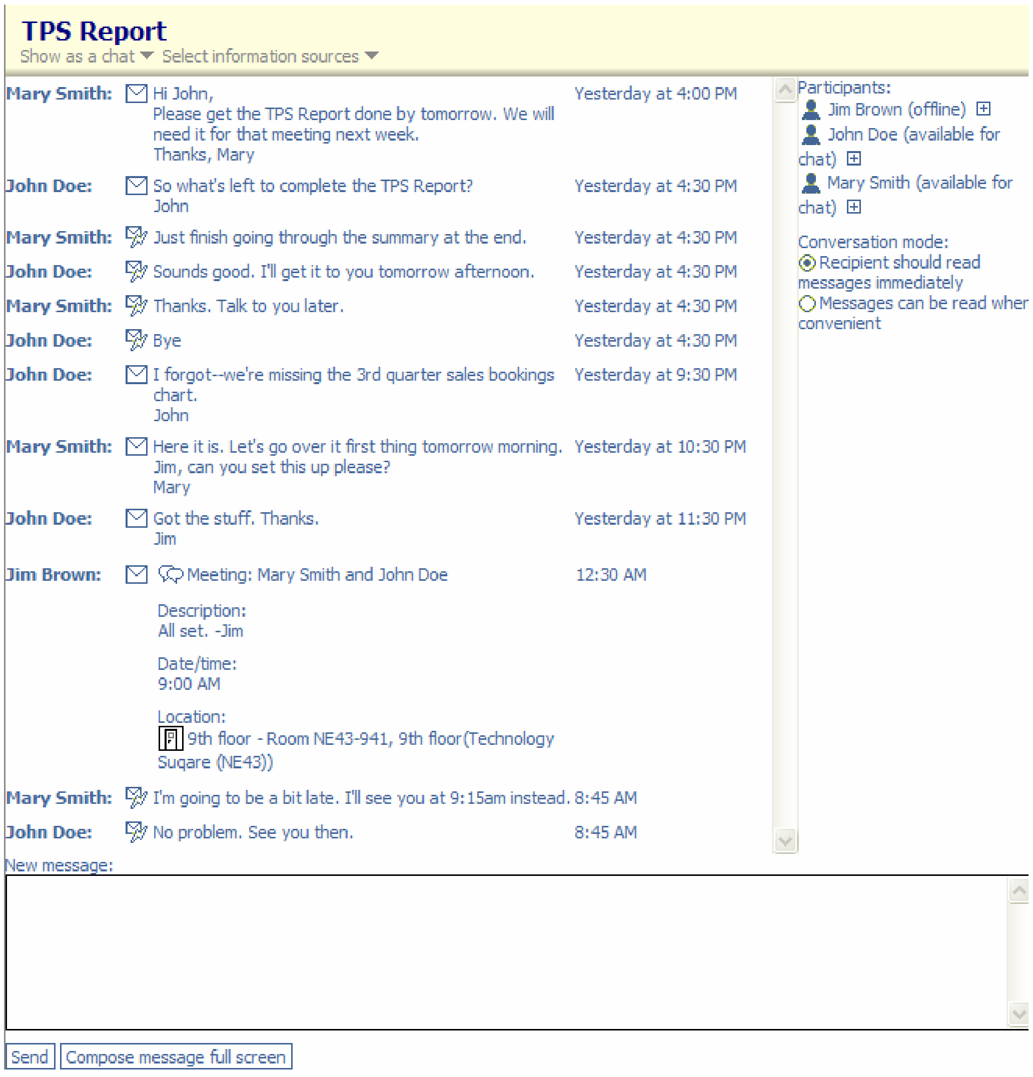
Built up from messages are higher level aggregations that model patterns of communication. We highlight two specific forms that pervade electronic communication, threads and conversations, while noting that these two forms can exist independently of each other and that other forms can be defined.

A thread is a connected subset of a collection of messages. Threads typically indicate a stream of messages exchanged between different parties on a very specific topic. Keeping track of threads is important because responses can sometimes be interpreted only in the context of other messages in the thread. Our concept of threading is indicated by the presence of the `msg:thread` property, linking a message to a `msg:Thread` object.

Threading is not directly supported by most e-mail protocols, but one method for constructing `msg:Thread` objects is to reconstruct threads based on `msg:inReplyTo` connections corresponding to the "in reply to" relationships present in e-mail and newsgroups. However, users often misuse the "Reply" feature of e-mail (e.g., sending a message to someone by locating the last message received from the person and clicking Reply) or fail to use it at all. Still, we feel that using `msg:inReplyTo` is a useful indicator of a thread, despite occasional human error. Perhaps providing a user interface that makes more prominent use of reply chains and gives users more control over how messages are sent will help to fix this problem.

Also important is the concept of a conversation. Like threads, conversations consist of a collection of messages, but the connection between messages in a conversation tends to be more loosely defined by a more generalized topic than those in a thread. Conversations in our ontology have type `msg:Discussion` and correspond to newsgroups, instant messaging conversations, and IRC chats.

In addition to being a collection of messages, conversations also maintain state in order to help facilitate changes in the interaction. A conversation resource keeps a record of the current participants in the conversation, as this list may change during the course of the conversation. Furthermore, whether the conversation is currently considered public or private or being held synchronously or asynchronously is recorded.



**TPS Report**

Show as a chat ▾ Select information sources ▾

| | | |
|---|---|---|
| Mary Smith: | Hi John,<br>Please get the TPS Report done by tomorrow. We will need it for that meeting next week.<br>Thanks, Mary | Yesterday at 4:00 PM |
| John Doe: | So what's left to complete the TPS Report?<br>John | Yesterday at 4:30 PM |
| Mary Smith: | Just finish going through the summary at the end. | Yesterday at 4:30 PM |
| John Doe: | Sounds good. I'll get it to you tomorrow afternoon. | Yesterday at 4:30 PM |
| Mary Smith: | Thanks. Talk to you later. | Yesterday at 4:30 PM |
| John Doe: | Bye | Yesterday at 4:30 PM |
| John Doe: | I forgot--we're missing the 3rd quarter sales bookings chart.<br>John | Yesterday at 9:30 PM |
| Mary Smith: | Here it is. Let's go over it first thing tomorrow morning.<br>Jim, can you set this up please?<br>Mary | Yesterday at 10:30 PM |
| John Doe: | Got the stuff. Thanks.<br>Jim | Yesterday at 11:30 PM |
| Jim Brown: | Meeting: Mary Smith and John Doe<br>Description:<br>All set. -Jim<br>Date/time:<br>9:00 AM<br>Location:<br>9th floor - Room NE43-941, 9th floor(Technology Sugare (NE43)) | 12:30 AM |
| Mary Smith: | I'm going to be a bit late. I'll see you at 9:15am instead. | 8:45 AM |
| John Doe: | No problem. See you then. | 8:45 AM |

Participants:
Jim Brown (offline)
John Doe (available for chat)
Mary Smith (available for chat)

Conversation mode:
[✓ Recipient should read messages immediately]
Messages can be read wher convenient

New message:

**Figure 3: Example conversation**

Depending on the protocol, conversations can sometimes serve as endpoints of messages. For example, messages in IRC chats and newsgroup postings are often not directed at any individual in particular but instead to the group, represented by the conversation object.

### 4.5 Annotations

In our model, annotations are simply messages with a well-defined topic resource. Like messages, the concept of being an annotation is completely detached from an object having other types, so any object can be made to be the body of an annotation. All of the previously described characteristics of messages apply equally to annotations; in particular, annotations can be replied to. Annotations have type `ann:Annotation` and the object being annotated connects to an annotation with the `ann:annotation` predicate.

The ontology presented here mainly addresses the notion of an annotation in a collaborative context. In such cases annotations are most often persisted into an annotation store. Annotation stores are similar to newsgroup servers in that they both contain groups of threaded messages loosely related by common topic spaces, and as such, annotation stores can be modeled as mail drivers in our model. However, our implementation of shared annotations makes no assumptions about the underlying store; a conversation held over an IRC channel makes for an equally suitable medium for holding annotations.

## 5. USER INTERFACE

Our unified messaging ontology gives us a means for integrating currently available modes of communication in an extensible fashion. However, for the user to realize the benefits of this ontology, the user interface must be carefully constructed as to capture the expressiveness of the underlying data model while preserving the benefits of the specific channels highlighted in Section 2.

### 5.1 Example scenario

To illustrate the architecture of our user interface paradigm, we will refer to the example conversation given in Figure 3. The scenario we pose features a dialog between John and Mary attempting to coordinate the completion of a report.

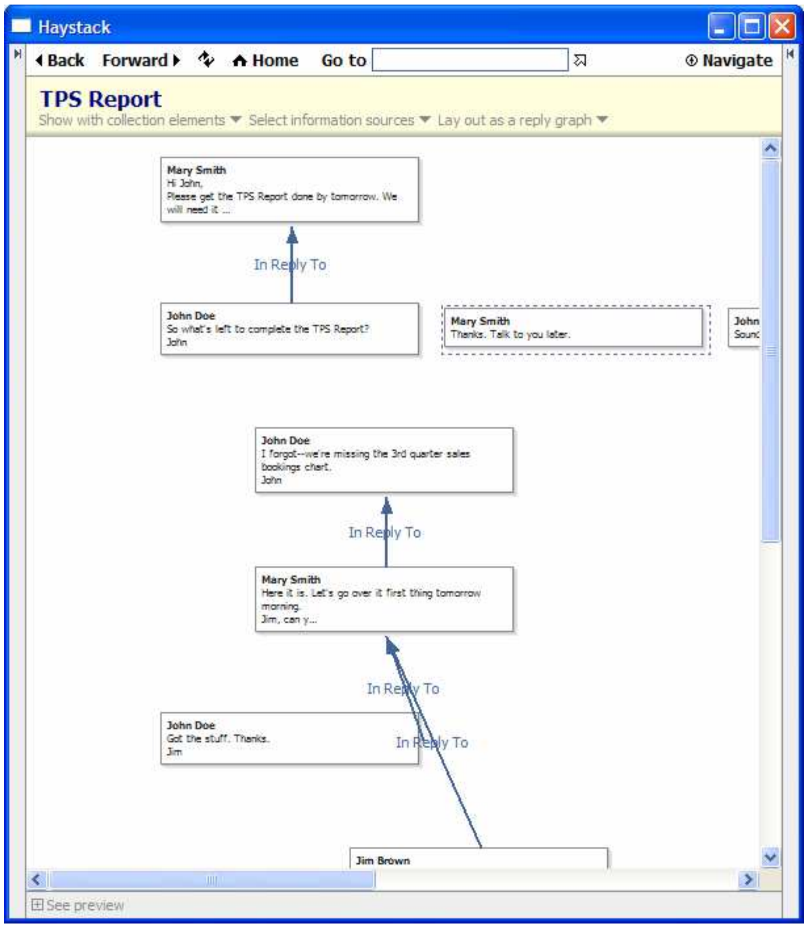
The following is a list of steps taken by Mary in this conversation:

1. Mary checks her calendar and notices that a key report is almost due. She selects "Send message" from the "Communicate/work with others" menu on the start pane. In the message editor she types up a message to John telling him to finish the report.
2. She receives a reply a few minutes later, where John has asked her about what is left to be done. She realizes that this conversation might require several turns. Mary decides to start tracking this conversation by selecting "View Discussion" from the context menu of John's reply.
3. In the conversation view she notices that John is online and changes the conversation mode to instant. She then has a short dialog with John explaining what needs to be finished.
4. That evening she checks her e-mail and notices a message from John telling her that an important chart is missing. She starts to type up a quick response when she notices that she needs to CC her secretary and attach a file. Mary clicks "Compose message full screen" in order to take advantage of the full screen message editor, which includes the advanced functionality she needs for this particular message.
5. In the morning she checks the conversation again and finds the meeting invitation sent by her secretary. Unfortunately, she is running late and sends John a quick message updating him of her status.

### 5.2 Conversation interfaces

It is useful to note that the example conversation interleaves both synchronous (instant messages) and asynchronous (e-mail) communication. Users can select the synchronicity when composing new messages by means of the Conversation Mode widgets. Also of note is the way messages of different types—here, textual and machine-processable structured messages (e.g., meeting requests, invitations, etc.)—can be combined within the same conversation.

The view of the conversation shown in Figure 3 arranges mes-



**Mary Smith**
Hi John,
Please get the TPS Report done by tomorrow. We will need it ...

**John Doe**
So what's left to complete the TPS Report?
John

**Mary Smith**
Thanks. Talk to you later.

**John**
Sounc

**John Doe**
I forgot--we're missing the 3rd quarter sales bookings chart.
John

**Mary Smith**
Here it is. Let's go over it first thing tomorrow morning.
Jim, can y...

**John Doe**
Got the stuff. Thanks.
Jim

**Jim Brown**

**Figure 4: Conversation displayed as a graph**

sages as a linear, temporal sequence. However, it is not difficult to imagine viewing the same conversation in a variety of other presentation styles, e.g., threaded messages, reply graphs, etc. In general, Haystack allows users to visualize data in different views depending on the context. For example, the same conversation can be visualized as a graph, as is shown in Figure 4.

Similarly, users can compose messages using a view that best suits their purposes. While participating in a chat, composition can be performed in-place. Messages composed in this context are sent in the context of the conversation; in other words, all such messages are automatically added to the conversation. Messages can also be composed in a full screen message editor, as is shown in Figure 5.

## 5.3 Reconstructing conversations

There are many ways users may choose to initiate communication with their colleagues. A user may know *a priori* that the message he or she is about to send will bring about an extended conversation. In this case the user can create the conversation explicitly by selecting “Start discussion” from the “Communicate/work with others” menu, akin to creating a private newsgroup or IRC channel. The discussion that is created is peer-to-peer in that it is maintained over instant messaging and e-mail protocols. Newsgroups and IRC channels can be supported with the same conversation user interface, highlighting the uniformity our system provides to communication.

However, in other situations, a user will send a quick informative message without expecting the numerous replies that ensue. At some point the resulting dialog may become important enough to warrant separate treatment as a conversation as modeled by our ontology. It is crucial that the user interface support *a posteriori* changes to the way a user organizes his or her messages and conversations.

When users are ready to start managing a stream of messages as a conversation in the user interface, he or she can select View Discussion from the context menu of a message to instruct the system to assemble all of the relevant messages into a conversation. Currently, the system uses the `msg:inReplyTo` relationships that exist between messages to locate the connected subgraph of the entire message corpus to identify the conversation.

In other cases, users may wish to collect messages that are loosely related into a conversation but that do not form a subgraph as defined above. Haystack allows users to drag and drop messages into the view of a conversation to indicate that the messages are a part of the conversation. Users can also add messages to conversations by treating the conversation as a category; the use of categories in Haystack will be discussed later.

## 5.4 Abstraction of identity

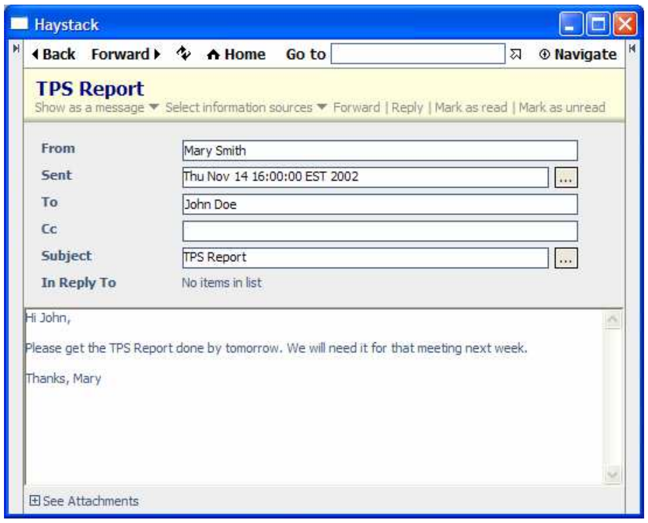
Our ontology gives users the ability to manage their contacts using a single identity, instead of the separate identities that are assigned on a per-protocol basis. As a result, it is natural to allow users to set up communication with their contacts regardless of protocol. Haystack exposes commands that allow the user to query for all messages to or from any contact from the contact’s context menu. Similarly, users can start conversations with or send messages to others through the same context menu. Refer to Figure 7 for a depiction of such a context menu.

## 5.5 Organizing messages

Given a system in which messages can be managed uniformly regardless of protocol or system, we are able to provide features to users that can help them to keep their various communications organized and that work over all types of messages. As Whittaker points out, e-mail (and similarly, other messaging systems) subjects the user to information overload, resulting in important messages not being attended to [5]. This drawback is primarily due to a combination of the high rate of information arrival, users lacking time or being unwilling to file away items and a lack of features that mitigate the effects of overload such as flexible message status tracking capabilities, reminder scheduling, semantic clustering of loosely related items and more powerful IR support.

The Categories pane in Haystack allows users to easily associate objects with one or more categories by presenting a list of checkboxes (see Figure 6). A check indicates that the currently viewed message belongs in the corresponding category. Haystack comes with a default list of categories, but this list can be easily customized by the user. In addition to being able to attach messages to conversations, users are also able to create categories such as “Letters from Friends” or “Favors” under which messages can be filed in order to help them locate important messages later. The use of checkboxes is suggestive of the system’s ability to allow items to be filed under multiple categories simultaneously. Previous work has suggested that such filing systems, which do not force users to choose among potentially overlapping categories, may be beneficial to users [15]. The system also provides default categories that typically apply to messages, e.g., “high importance”, “FYI”, etc.

Furthermore, as users classify messages that arrive into the system, they are providing implicit descriptions for their categories by the presence of the messages that these categories contain. Haystack supports an extensive agent infrastructure that allows its RDF repository to be used as a blackboard, creating the opportunity for agents to apply classification and clustering algorithms to “learn” why certain messages are filed into specific categories [2]. In the future, support is planned that will allow Haystack to sug-


Haystack
Back
Forward
Home
Go to
Navigate
Show as a message
Select information sources
Forward | Reply | Mark as read | Mark as unread
See Attachments

TPS Report

| | |
|---|---|
| From | Mary Smith |
| Sent | Thu Nov 14 16:00:00 EST 2002 |
| To | John Doe |
| Cc | |
| Subject | TPS Report |
| In Reply To | No items in list |

Hi John,

Please get the TPS Report done by tomorrow. We will need it for that meeting next week.

Thanks, Mary

**Figure 5: Full screen message editor**

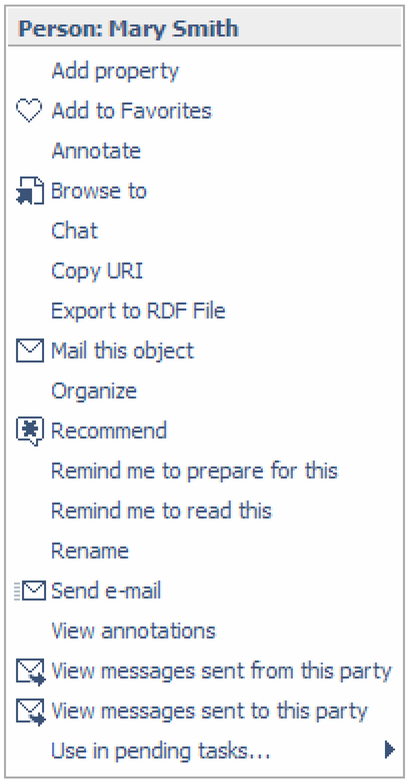


**Figure 7: Context menu for a person**

gest categories for messages that it believes best characterize the message, based on the results of these learning algorithms.

## 6. FUTURE WORK

Our prototype built on Haystack represents an evolutionary step in terms of providing a single, unified interface for supporting interpersonal and group communication. However, much work remains to be done in order to produce an intuitive system that can be employed by users on a daily basis. Although we have asserted a unified messaging model to be useful, user studies are required to understand whether users prefer a unified user interface or would rather keep the distinction present in current messaging paradigms. Similarly, support exists for hosting threaded discussions around annotated documents, but further exploration into the connections between messaging and group annotation is warranted.

In terms of the implementation, our initial prototype includes support for POP3, SMTP, Jabber instant messaging, and a lightweight peer-to-peer messaging protocol built on top of JXTA. Although we have demonstrated that the ontology can support a myriad of different messaging protocols, we wish to demonstrate this by incorporating support for the other messaging systems discussed in this paper, e.g., NNTP, IRC.

Finally, we hope to further exploit RDF's capability to encode arbitrary metadata in order to realize other benefits of the Semantic Web. By employing richer forms of categorization and semantic markup in order to better describe the nature of messages being sent, recipients will be able to better manage their information influx. For example, if a piece of information being sent has already been characterized on the sender's end, the recipients' systems may be able to automatically file the information into the proper categories or process requests embedded within the infor-

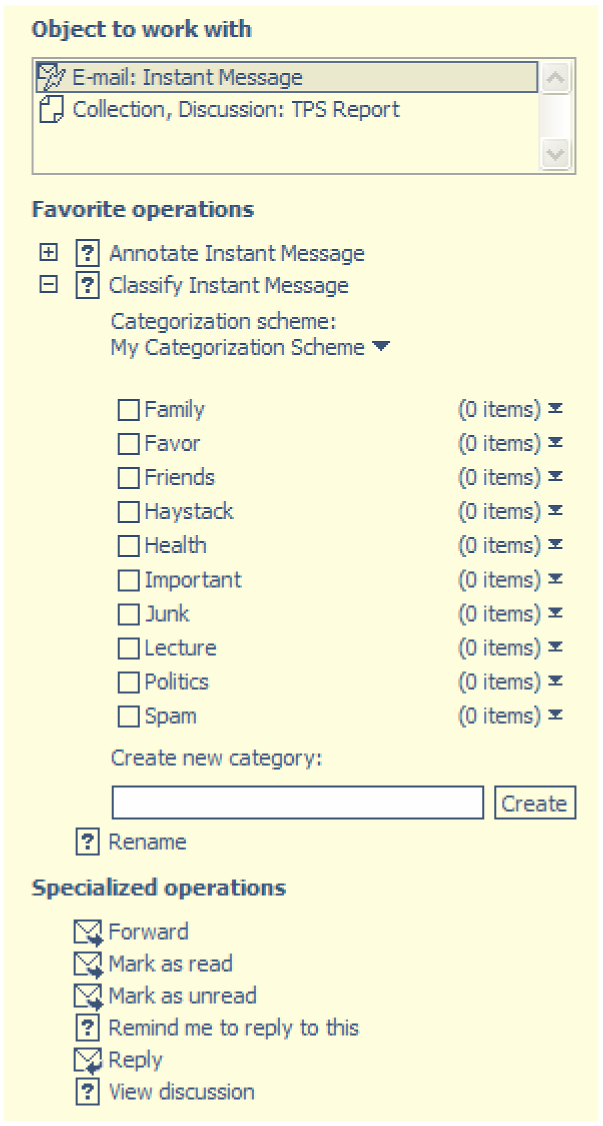


**Figure 6: Categorization pane**

mation, such as meeting invitations. Also, it is not inconceivable to imagine the user's system serving as a "gatekeeper" for the user, managing a user's inbox by prioritizing received information for the user's consumption.

## 7. ACKNOWLEDGMENTS

This work was supported by the MIT-NTT collaboration, the MIT Oxygen project, and IBM.

## 8. REFERENCES

[1] Huynh, D., Karger, D., and Quan, D. (2002). "Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF." Semantic Web Workshop, The Eleventh World Wide Web Conference 2002 (WWW2002). http://haystack.lcs.mit.edu/papers/sww02.pdf.

[2] Quan, D., Huynh, D., and Karger, D. "Haystack: A Platform for Creating, Manipulating and Visualizing Information on the Semantic Web." In submission for the World Wide Web Conference 2003 (WWW2003).

[3] Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[4] Berners-Lee, T., Hendler, J., and Lassila, O. "The Semantic Web." Scientific American, May 2001.

[5] Whittaker, S. and Sidner, C. "E-mail Overload: Exploring Personal Information Management of E-mail." Proceedings of CHI 96: Human Factors in Computing Systems.

[6] Smith, M., Cadiz, J., and Burkhalter, B. "Conversation Trees and Threaded Chats." CSCW '00.

[7] Nardi, B., Whittaker, S., and Bradner, E. "Interaction and Outeraction: Instant Messaging in Action." CSCW '00.

[8] Whittaker, S., Terveen, L., Hill, W., and Cherny, L. "The Dynamics of Mass Interaction." CSCW 98.

[9] Voida, A., Nestetter, W., and Mynatt, E. "When Conversations Collide: The Tensions of Instant Messaging Attributed." CHI 2002.

[10] Berners-Lee, T. Primer: Getting into RDF & Semantic Web using N3. http://www.w3.org/2000/10/swap/Primer.html.

[11] Whittaker, S. "Talking to Strangers: An evaluation of the Factors Affecting Electronic Collaboration." CSCW 1996.

[12] Cadiz, J., Gupta, A., and Grudin, J. "Using Web Annotations for Asynchronous Collaboration Around Documents." CSCW 2000.

[13] Kahan, J. and Koivunen, M. "Annotea: An Open RDF Infrastructure for Shared Web Annotations." WWW10, 2001.

[14] Quan, D., Lin, J., Katz, B., and Karger, D. "Sticky Notes for the Semantic Web." IUI 2003.

[15] Lansdale, M. "The Psychology of Personal Information Management." Applied Ergonomics, vol. 19, no. 1, 1988, pages 55–66.

[16] Box, D., Ehnebuske, D., Kavivaya, G., et al. SOAP: Simple Object Access Protocol. http://msdn.microsoft.com/library/en-us/dnsoapsp/html/soapspec.asp.