The Message without the Medium: Unifying Modern Messaging Paradigms through the Semantic Web

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

David R. Karger
MIT CSAIL
200 Technology Square
Cambridge, MA 02139 USA
karger@theory.lcs.mit.edu

Dennis Quan
IBM Watson Research Center
1 Rogers Street
Cambridge, MA 02142 USA
dennisq@us.ibm.com

ABSTRACT
Since its inception, the Internet has been a hotbed of successful communications channels, starting off with e-mail, Internet Relay Chat and Usenet newsgroups and more recently adding weblogs, instant messaging, and news feeds. These systems have been developed quite independently over the past half century and continue to be extended with new functionality that addresses the broadening needs of their users and supports the full range of semantic expression. Stepping back, however, we observe that having a variety of messaging frameworks creates significant problems for users when attempting to manage and collate messages on a single topic or context that may be discussed via multiple media. We posit that no message should be constrained in this way by its medium. As it is, messaging applications are slowly converging in their functionalities. We show that a unified approach to messaging can be achieved in a single step through appropriate use of the RDF, a Semantic Web technology, as a data model. We further exploit this data model to develop appropriate user interface elements that allow aggregation of messages across protocols, and discuss the benefits that arise from such a scenario.

Categories and Subject Descriptors
H.4.3 [**Information Systems Applications**]: Communications Applications – *electronic mail, information browsers, bulletin boards, Computer conferencing.*

General Terms
Management, Design, Human Factors, Standardization

Keywords
E-mail, instant messaging, IRC, newsgroups, annotation, web logs, blogs, news feeds, RSS, wiki, user interface, messaging, RDF, Semantic Web, information management

1. INTRODUCTION
One of the greatest accomplishments of the Internet has been to enable communication in various forms. E-mail, instant messaging (IM), Internet Relay Chat (IRC), Usenet newsgroups, weblogs, and RSS feeds all provide channels through which one individual or group can convey information to another. Arguably, this plethora of protocols was developed because of the many variations in how people use messaging.

WWW 2004, May 17-22, 2004, New York, NY USA.
ACM xxx.

These systems have been developed quite independently over the past half century and continue to be extended with new functionality that addresses the broadening needs of their users and supports the full range of semantic expression. Stepping back, however, we observe that having a variety of messaging frameworks creates significant problems for the user in maintaining context across messaging media. As a result, some efforts to address this problem have begun but are cosmetic at best (e.g., Microsoft and Apple) or only deal with a subset of it (e.g., ReMail) [17][19].

Whereas an attempt at unifying messaging via another general solution is certainly possible, we feel that the Semantic Web presents a good opportunity, and is the right forum to begin to address the problem. One of the core themes of the Semantic Web is integrating and enabling interoperation between disparate systems. Recent research has shown the benefits of using RDF to enable integration in areas ranging from Semantic Web Services to bioinformatics [20][21]. Furthermore, the Semantic Web, true to its name, has benefited the original Web in tying together related Web pages by means of ontologically-classified metadata, as recent efforts in large scale mining have demonstrated [22]. Still, we feel that one critical application of the Internet has yet to benefit from this revolution: communication.

1.1 Motivation
The existence of multiple, independent messaging systems presents users with various problems. For example, in choosing a messaging medium, the sender is restricting him or herself and the recipient to a particular subset of messaging functionalities; if he or she chooses to use IM to discuss a work-related project, participants cannot file it away in an appropriate IM folder. At best, the user can save it to a file and relocate the file to an aptly named directory in the file system. Of course problems in disparate functionality can be overcome to a great extent by "feature creep"—the gradual addition of enhanced functionality in the appropriate messaging application, which while unanticipated, is eventually deemed to be useful. A good example of this can be found in IM clients: whereas early clients were limited in functionality to text messaging, more recent ones support sharing images, sharing documents, videoconferencing, and even digital whiteboard-based interactive collaboration. Similarly, 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. Furthermore, newsgroup readers and e-mail clients both have threaded message views and different mechanisms for filtering out messages from specific people. Finally, we can soon expect sophisticated anti-spam capabilities in IM as the problem worsens in that domain, similar to e-mail.

However, the problems of having multiple messaging media do not end by supplying parity in functionality. They become even more significant when many messages are exchanged to form a conversational thread. A user may initiate a conversation over IM, expecting it to be short. If the discussion becomes more extensive, bringing in more participants and generating lengthy e-mails or chat sessions, there is no convenient way (beyond cutting and pasting) to bind the early, IM portion of the discussion to the later, e-mail or chat-based portions; if a message is sent by IM, one cannot file it in an e-mail folder. Moreover, it is not possible to reconstruct a threaded view of the initial instant messages and the e-mail responses to them. The fact that media other than e-mail are used for work-related communication is supported by Isaacs et al., who report that contrary to popular belief, the major use of IM in the workplace is for "complex work discussions," rather than coordination or simple question answering [16]. Similarly, Handel and Herbsleb report that chat in the workplace is primarily used for work-related discussions [18].

Practically speaking, when a single conversation is often "smeared" across different media and channels, both functional disparity and segmented storage exacerbate the problem of user context maintenance. If messages exchanged via different messaging media need to be or are semantically related, then the combination of differences in stores, formats and data models and the set of supported operations for a given medium can get in the way of using them effectively. For example, how can a quick IM conversation pertaining to a particular design element of some software be captured in the project's log for purposes of traceability and provenance? Alternatively, how can knowledge gleaned from a newsgroup or RSS feed and the subsequent discussion with the author of the information be captured in a project's historical log?

Even for users that have the discipline to use the same messaging medium for all semantically related information, the problem would still insidiously persist because not all semantics are known *a priori*. That is, some information may become related to other information only in retrospect. How then should the two be related so that they can be accessed uniformly and simultaneously when working on the topic? Does the application permit the user the flexibility of assigning semantics and context? The situation becomes more complex (and common) when considering the problem across multiple messaging media.

Thus, from a user's perspective, the selection of the medium to use is a situational and convenience decision as to how to structure the message exchange mechanics. The medium content is orthogonal to the medium selection. The duplication of features in order to increase parity in content capabilities of various media is a symptom of the problem of needing to be able to use different media for communication as the situation warrants without sacrificing expressiblity. But, feature duplication is not completely successful at resolving the problem, e.g. why can a meeting request in MS Outlook be sent over e-mail, but not over IM? It is a symptom of the decoupled nature between message (content/semantics) and medium (messaging channel) and only encourages practices that exacerbate the problem of context maintenance across media.

Such a problem is to be expected as multiple systems are all being used simultaneously and providing different styles of solutions to the same fundamental problem of interpersonal and group communication. While in the beginning, the different communication channels crystallized pieces of functionality specific to key activities along with the associated infrastructure (e.g., addressing and authentication) and then were enhanced with additional features, users have now grown more reliant on using these systems interchangeably and are now bumping against the limitations of the abstractions since the activities no longer necessarily respect the medium boundaries [14][17].

1.2 Approach

We posit that no message should be constrained in this way by its medium. Instead, a user should be able to look at any message in any way that is natural, and apply to the message any natural message-handling action, regardless of the particular message medium—in effect, unify messaging to treat all message types as equivalent semantic entities via a unified data model. 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 that decouples messaging media, which determine the messaging mechanics/techniques, from messaging content/semantics—the aspect of greater import to the user. People use different media to take advantage of the differences in the mechanics of communications, e.g. addressability and timing of communications. These should be decoupled from the semantics of communication as far as the user is concerned. We apply RDF, a key Semantic Web Technology, in unifying the data model for various messaging paradigms.

By unifying the messaging paradigms, we realize a number of immediate benefits to various parties, as well as allowing messaging to remain open to future developments. Once the existing systems are unified under a common model, we can enhance all forms of messaging by incorporating features that are currently present only for specific messaging paradigms. A unified approach will further enable us to allow the user better control of context maintenance since all types of messages will be first class entities that can be arbitrarily collected to create a context. Also, a user will be able to retrospectively create contexts that are currently impossible. In addition, it will further ease application and user interface development and aid UI researchers seeking to allow the user to better manage his/her information. Not only will current messaging paradigms be captured by the data model, but the RDF based data model will be robust enough to allow for integrating new messaging paradigms in the future. Furthermore, using a unified data model based on RDF and the Semantic Web for messaging supplies an incredible corpus of information that is interoperable with the Semantic Web, and hence open to processing by Semantic Web agents. Finally, expressing a unified model of messaging via RDF, we render messaging amenable to the exciting possibility of *Semantic Messaging* whereby the message bodies are semantically marked up based on well known ontologies. As a result, they would be open to automated processing, yielding new avenues for enhancing personal productivity through more sophisticated CSCW, thereby further leveraging the promise of the Semantic Web.

We show that such a unified approach to messaging can be achieved through appropriate use of RDF, a Semantic Web technology, as a data model and appropriately designed user interface elements. In Section 4.1, we sketch a unified ontology representing all the different message types discussed above. The details of the ontology are not important; rather, our goal is to highlight the fact that all the above messaging frameworks share substantial conceptual structure, reflecting ideas such as sender, recipient, subject, reply to, and so on. It is these abstract properties, rather than the specific messaging protocol, that really determine the role of a message. In Section 4.2, we describe a single running exam-

ple that illustrates a unified tool for handling messages arriving via some of the protocols listed above. Our tool is built within the Haystack system, a unified information management environment [1]. It exploits the common messaging ontology to present all incoming information to its user in convenient ways. In Section 5, we discuss the benefits that arise from aggregating all the messaging protocols.

2. RELATED WORK

In this section, we discuss related work that identifies and attempts to ameliorate problems similar to those resulting from multiple messaging media but in the context of e-mail, thereby supporting our motivational assertions. Next, we address previous attempts at attacking the problem across multiple messaging paradigms. Finally, we conclude with a short survey of common messaging paradigms as a basis for informing our messaging unification work.

2.1 Task Management in E-mail

Maintaining the semantic boundaries of information from a user's perspective is a fundamental problem, and does not require the complexities of multiple messaging media to manifest itself. From a user's perspective, e-mail supports important abstractions encapsulated by a semantic boundary: tasks and archives [4]. Given that "users have co-opted this flexible application [email] as a critical task management resource," a significant amount of recent research interest has focused on improving e-mail as a task management application [14]. In a field study designed to inform the design of TaskMaster, Bellotti et al. uncovered that the primary problem users experienced in using e-mail for task management resided in being able to get "a task oriented overview, at a glance, rather than scrolling around inspecting folders" and "collating related items (e.g., an extended thread or responses to a survey) and associated files and links" [14]. In general, the problem occurred whenever information required within a context was not easily available, e.g. it scrolled out of view due to unrelated messages filling the inbox, or was available in a separate list such as MS-Outlook's contacts, outbox, calendar or to-do items, etc. Furthermore, they observed that "threads of activity in email do not always correspond to straightforward message threads," and hence it is important to allow "users [to] fine-tune the contents" and "users need to cut across application boundaries in their work…[and]…include items from their desktop or useful links that have never been sent in email" into the task related collection. As a result of the study, Bellotti et al. developed the TaskMaster system, which takes a purely UI approach at improving e-mail for task management. A related commercial effort by Kubi Software seeks to improve task management by developing project contexts that co-locate relevant items such as contacts, messages and documents in MS Outlook and Lotus Notes e-mail clients [15].

Thus, we can come to appreciate the importance of context or semantic boundary from a user's perspective. Furthermore, it is not too difficult to see that such a boundary may be needed for abstractions other than tasks and how other means of exchanging information incorporating "a variety of types of media" similarly support task management [14]. As these media capture additional knowledge artifacts, the content (and context) management problems across messaging paradigms we outlined earlier will increasingly emerge.

2.2 Previous Work in Unifying Messaging

Recognizing the importance of other messaging media in the workplace, several attempts have begun to attempt to combine the two. For example, the Microsoft Outlook e-mail client now supports awareness tracking by appropriately coloring the contact/sender/receiver icon and IM initiation. However, the initiated IM session goes via the MSN Messenger application, and any persistence that is possible, is only allowed via it. Thus, the IM session cannot be persisted in the context of the mail messages. Apple's iChat is similarly, superficially, integrated with the mail application and address book.

A definite improvement over this state of affairs is the ReMail system developed under the Collaborative User Experience Project at IBM [17]. ReMail allows not just awareness tracking of colleagues through e-mail, but also starting an instant message conversation simply by clicking on the person to initiate a chat session and persisting it in the context of the ongoing thread (possibly e-mail) of conversation. Thus, a user may change between IM and e-mail as the urgency of the *situation* warrants rather than whether he or she will eventually need to capture the conversation in a particular manner. Interestingly, ReMail also allows the user to annotate his/her e-mail for future reference to self or others, e.g. secretary. ReMail seems to have identified the same problem as us and implemented a reasonably complete and robust solution, it has done so only for IM and e-mail. Also, it is unclear whether the enhanced functionality has been accompanied by a change in data model, or IM and e-mail have been integrated over still disjoint infrastructures. Thus, unlike our promotion of a more comprehensive solution for messaging in general via data model integration of disparate messaging systems that supports future extensibility, it has taken an incremental, organic approach similar to "feature creep" that will add additional functionality to messaging systems as the need becomes clear.

3. CURRENT COMMUNICATION SYSTEMS

Having demonstrated that the notion of semantic boundaries and context in messaging are important from a user's point of view and current attempts to provide integrated support for it across IM and e-mail, we now turn our attention to considering the "big picture:" unifying various messaging paradigms in order to support this notion universally. As such, we survey some popular existing digital communications mechanisms, their particular properties, usage niche and existing problems in order to understand the nature of the communications they facilitate, and how it can be captured succinctly.

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 [4]. This is primarily due to the e-mail inbox being capable of maintaining context for related messages, simplifying information availability by co-locating 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.

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 [5]. 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.

Newsgroups. Newsgroups comprise perhaps the largest online communities that have resulted from the proliferation of the Internet [6]. Whittaker's findings on group discussion seem equally applicable to newsgroups in general [8]. 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, a minority of newsgroup users contribute a majority 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 [8]. 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. It would be useful to be able to capture relevant portions of news in other contexts.

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 [8]. 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.

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 [9]. Also of interest are recently developed annotation systems that permit both metadata and textual messages to be specified [10] [11].

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"). Although developed as a framework for broadcasting, it is not difficult to view them as ones for messaging in general. 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.

Wikis. Wikis are a recent phenomenon in collaboration on the web that allow users to not just be passive recipients of information on the world wide web, but also actively be able to edit it as website. Thus, it conflates the traditional notions of webmaster and audience for a website. We consider it here because inherent in all collaborative systems is the notion of messaging. Thus, we may consider Wikis as a means of messaging. Much like annotations or newsgroups, they are messages to an unknown audience. However, unlike newsgroups and annotations, the sender/editor is generally anonymous.

3.1.1 Axes in the Messaging Space

We use our survey of messaging paradigms to realize that despite their ostensible differences, all communication systems can be placed in context and compared by considering a handful of useful partitioning criteria. We juxtapose communication paradigms with respect to two major partitioning criteria in Table 1. Other axes for partitioning the messaging space are also discussed. However, they result in a less clear cut segmentation of the space as the same application is not restricted to a particular place on an axis or are closely correlated with values on other axes. We discuss some of the more common axes here, while realizing there are others. We then use these findings as a starting point for our integration work.

3.1.1.1 Public Versus Private

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

3.1.1.2 Synchronous Versus Asynchronous

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. 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. [7].

3.1.1.3 Persistent Versus Ephemeral

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 persist automatically. 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 [8]. On the other hand, short and to the point messages characterize synchronous communication, with persistence usually being an explicitly specified option due to the way these systems have evolved.

3.1.1.4 Unidirectional Versus Bidirectional

Messaging systems can also be loosely categorized as being unidirectional or bidirectional. Simply put, bidirectional messaging generally requires participation from multiple parties and the sender and recipient is clearly recognizable and addressable, e.g. e-mail, IM. In unidirectional messaging, content is just "put out there" and the audience is not always required or able to respond. Examples of this include newsgroups, RSS feeds and Wikis. However, it is not uncommon to have directed messages soliciting a reply from particular users in newsgroups either. Finally, traditional bidirectional messaging systems such as e-mail can be co-opted for unidirectional messaging, e.g. periodic newsletters, and can specify a reply address.

4. APPROACH

In developing a unified framework for messaging, our approach aims to preserve the existing communication capabilities supported by the Internet. Allowing the messages to be co-located would be a step in the right direction, although not strictly necessary. More importantly, multiple message formats would have to be understandable and transparent to the messaging application. However, even that would leave the messaging application (e.g. MS Outlook) that understood multiple information entities and/or message types in the difficult position of not being flexible enough to easily assimilate new message types.

As such, we believe the data model should be unified to allow treating all message types as just messages; first class entities with respect to each other. From a user's perspective they already are a unified data model – sometimes users want to see just messages as units of communication pertaining to a single topic, and hence UIs should treat the underlying information as having a uniform data model. Furthermore, our survey of various communications mechanisms shows that all messages share (or can be modeled to share) the same notions, such as sender, receiver, body, synchronicity, persistence, privacy, etc. Also, many UI enhancements on which measurably improving the user experience will heavily rely, will require an expressive and extensible supporting data model as hinted at by [14] in their primarily UI based approach at enhancing e-mail. Otherwise, the enhancements will be merely cosmetic. Finally, the only way to effectively support greater, and more uniform UI functionality as witnessed by the feature creep in messaging applications is by supporting it through the data model. Instead of writing separate user interfaces for each message type, we can also simplify the user (and developer) experience by developing consistent user interfaces based on a single data model. Thus, our work begins with unifying the data model to allow support for multiple messaging paradigms, while still capturing the inherent advantages of each. Thereafter, we present a sample interface developed using this data model.

We have chosen to base our approach to unification of messaging paradigms by specifying our data model declaratively via a well defined ontology for messaging using the Resource Description Framework (RDF), a Semantic Web technology for integrating disparate systems and data together [3]. The ontology is understood and complemented by related driver code that translates the sending and receiving of different message types to the appropriate underlying protocol. RDF is easily expressible in a portable format for describing semantic networks or labeled directed graphs [2].

Table 1: Existing messaging systems

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

This decision serves our goal well on several fronts. First RDF supports a semi-structured modeling scheme that allows us the flexibility to not only capture the core aspects of messaging, but also the unique characteristics of different messaging paradigms (e.g. the notion of presence awareness does not exist in asynchronous messaging, but both synchronous and asynchronous messages have senders and receivers). Thus we can model a variety of concepts, from annotations to news feeds. Next, RDF also allows us to capture entities that can simultaneously have multiple types, e.g. a message can be both a meeting request and either an e-mail or instant message. Also, the RDF data model does not restrict the modeling to a strict message type hierarchy, thereby supporting future extensibility in integrating new communication paradigms. Finally, the RDF model enables exciting possibilities with semantic messaging that we discuss further in future work.

Our objective in developing the data model and sample user interface is not to show the perfect approach to unifying messaging, but rather to make the case that doing so using RDF is possible, and has not just significant advantages over the status quo not, but also provides a tangible application and a source for critical information mass needed for realizing the vision of the Semantic Web.

To fulfill the RDF data model, we are building support for unified messaging into Haystack, an information management project at the Computer Science and Artificial Intelligence Laboratory at MIT. 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 viewing and navigating the various kinds of information found in the user's corpus.

4.1 Unifying the Data Model

In this section we discuss the various elements of our messaging ontology. **Figure 1** depicts the different elements of this ontology and how they interrelate by means of an example.

In order to apply RDF to the problem at hand, we give ontological specifications of how to represent people, messages and higher level semantic groupings of messages, e.g. discussions, threads,

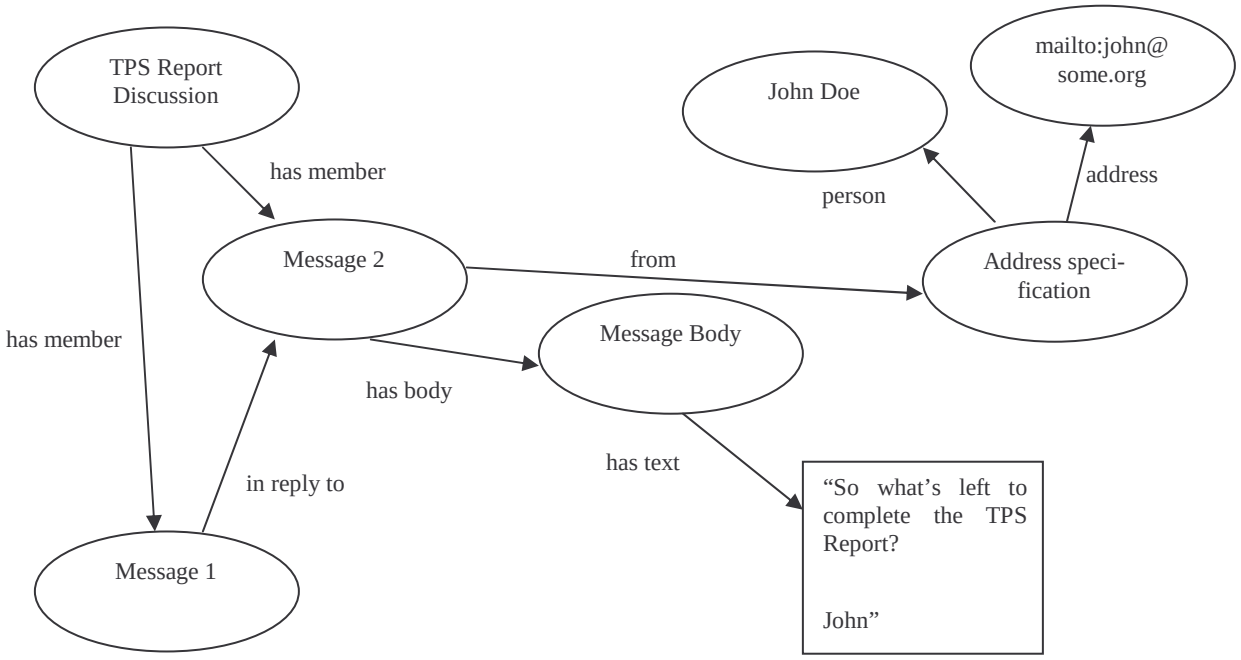

Figure 1: Messages modeled according to our ontology

by defining a set of classes and properties (also called predicates). 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 [4][5]. We avoid detailed discussion of individual message type enhancements on the data model for simplicity; they can similarly be modeled (e.g., an `onlineStatus` property can be associated with a person to track his/her online status for instant messaging). We realize the ontology by related driver code that abstracts away the complexities of the various underlying protocol implementations and allows the model and interface to address messaging in a user-centric manner so that many of the deficiencies in existing systems can be addressed at the root while simultaneously facilitating uniform functionality.

4.1.1 Identity and Addressing

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. 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.

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 (e.g., multiple e-mail identifiers, IM accounts, etc.), or several people may be sharing the same address. 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 `Person` class. Recipients and senders are specified by instances of the `AddressSpecification` class. Address specifications can specify either a specific address or a person resource, or both. People can be associated with addresses with the `hasAddress` predicate.

A few clarifications on identity and addressing are warranted. Although we have mentioned identity in the context of a person for simplicity, it is important to realize that many times, it is not the person, but the role that is the target of a message (e.g., sending a message to the webmaster), and hence we expand our notion of identity to include both person-based and role-based identities. Also, we draw a distinction between the notion of an audience and an address. Whereas a message is eventually always directed at an address, the audience is not necessarily defined. For example, a message may be directed to a newsgroup or wiki, but the audience depends on the dynamic subscriber list of the newsgroup or frequent (and possibly unknown) visitors to the wiki web. Although an address generally implies a well known audience, this may not always be the case. We only attempt to model the notion of address in our current work.

4.1.2 Messages

In order to incorporate the various forms of messaging available, we define the class `Message` in a very general manner. In our system a message is a unit of expressive communication transported from senders to 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.

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

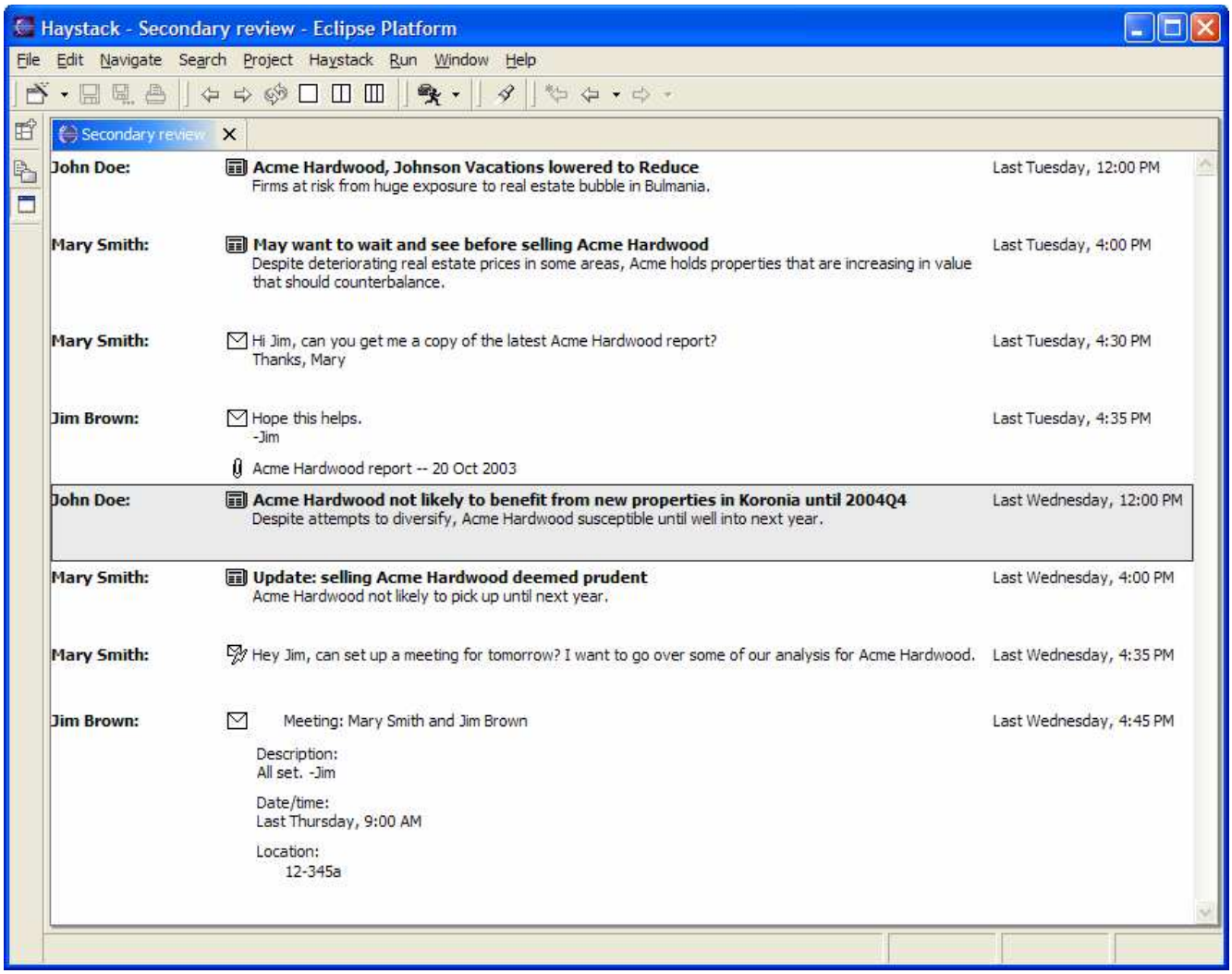

Figure 2: Example conversation

financial statements. Therefore, we define a message as a resource 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.

4.1.3 Conversations

Built up from messages are higher level aggregations that model patterns of communication or define other user specified contexts. We highlight conversations (i.e., user-defined contexts) in this subsection.

Like message threads, conversations consist of a collection of messages, but the connection between messages in a conversation is defined by the user and tends to be more loosely defined by a more generalized topic/context than those in a thread (and hence not necessarily based on `inReplyTo` connections). Thus, whereas the threads can be used as a starting point for a conversation, the user can add other threads to the conversation that are relevant or remove unrelated messages (e.g., when someone sends a message to another by locating the last message received from that person and clicking Reply).

4.1.4 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 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.

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 [12]. 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. Finally, messaging drivers for protocols that support particular properties, e.g., awareness of presence, are responsible for keeping these properties up to date in the RDF store.

4.2 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 systems highlighted earlier.

To illustrate how our user interface takes advantage of the data model, we will refer to the example conversation given in Figure 2. The mocked-up scenario we pose features a dialog between two stock analysts working at the same brokerage, which uses Web logs internally to keep fund managers and other personnel informed. John (real estate market analysis) and Mary (general analyst) are attempting to resolve some differences in their outlook for a particular stock that they are both watching.

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

1. Mary checks her inbox and notices that John has updated his blog on the real estate industry outlook. She notices that he has downgraded Acme Hardwood because he feels the real estate market will be going down according to a cyclical trend. However, her research has indicated that Acme Hardwood has just opened an office in Koronia, where real estate is booming. Hence, she feels that fund managers would be ill advised to follow John's recommendation. She refers to his message, and posts a correction in her blog.
2. John notices a reply from Mary and posts an update to his blog indicating that whereas it is true that Acme Hardwood has opened an office in Koronia, it will not be fully operational for another year. Thus, his recommendation stands.
3. Mary is concerned that her data might be inaccurate and asks for a company report from her colleague Jim in the research department.
4. Mary receives data capturing key company measures from which she extracts only those that are forward looking. She realizes that she has indeed missed the fact that the office will be operational next year. She replies to John's blog update and issues a correction.
5. Mary creates a new conversation ("Secondary Review") and sends a meeting request via instant message to Jim in its context, in order to hold a meeting to review the outlooks for all companies she oversees.
6. Jim's and Mary's meeting manager agents negotiate a free time, and book the appropriate room with the facilities agent for the meeting. Mary's agent notifies her she has a meeting with him at 9am Thursday.
7. Mary then adds the previous thread of conversation with John and her earlier messages with Jim to this conversation in order to maintain proper context for the new conversation.

This example illustrates several interesting capabilities that are presently not possible with existing messaging systems. The example starts out with Mary receiving a message that encapsulates the incremental change that John made to his blog. Whereas current changes supported by blogs overwrite previous results, it would certainly be nice to get incremental updates to them as messages if they are being closely followed as in this case. Next, Mary, is capable of responding to a blog message via another blog message. (Literally, an in-reply-to property is placed into the blog's RDF/RSS 1.0 file on the server.) This too would be intuitive from the user's point of view in some cases. She sends the reply both to her blog and John. John responds similarly. Next, Mary issues a request to the research wing of the firm, and receives the requested data semantically marked up so that she can appropriately query it. Mary then replies to John's blog entry with a different kind of message interleaved in the thread (an e-mail), but one that is treated as any other message. Also, Mary can create a new context, "Secondary Review" and send a meeting request within it via IM. Thus, she is able to use an instant modality of communication without sacrificing semantic expressibility as well as create a context *a priori*. Furthermore, since the message is semantically encoded, it can be automatically responded to by Jim's agent. The hidden negotiation between the agents is logged as a thread but only its final message is to Mary and Jim, confirming the appointment. It can be accessed if needed by Mary. Finally, Mary can retrospectively add the discussion with John to the current conversation using drag and drop to provide better context for it. Users can also add messages to conversations by treating the conversation as a category.

Note that the two options given in our interface are extremes of a more general spectrum; one could imagine the sender specifying a particular "read by" date and/or time. 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.

5. DISCUSSION

Our work discussed above presents a solution to a very real problem by employing RDF as a central Semantic Web technology that can be used to better model communication systems as messaging paradigms. As a result, the solution exhibits a number of desirable characteristics for various parties that would otherwise have not been possible. Also, although the problem could be solved without explicitly resorting to RDF and the Semantic Web, we feel that electronic communication comprises a significant portion of not just Web traffic, but Internet traffic as well. Thus, it would behoove a solution to the problem to stay close and interoperable with the future direction of the World Wide Web and Internet. Also, applying RDF to this problem may allow us to bootstrap the Semantic Web by building into it an engine that continuously generates semantically marked up information. Using RDF in our solution brings allows it to be forwards compatible and one step closer to realizing the potential of the Semantic Web.

Developers benefit from a unified data model for messaging as it allows them to quickly and easily support multiple messaging paradigms as well as rapidly incorporate new ones. Furthermore, the expressive power facilitated by a *single, flexible* data model, such as RDF, allows them to provide users with functionality not possible heretofore, e.g., capturing and co-locating content regardless of transmission protocol, a step towards relieving information overload. Another artifact of our generalized treatment of messages is that it provides common user interface functionality across all messaging paradigms [13]. For example, bodies of messages can be spellchecked in e-mail clients but not in IM clients; worse yet, e-mail clients support spell checking for the body but not for the subject of the message. With a uniform data model, both the body and the subject of a message can be spellchecked regardless of message type because they are both text fields. Fi-

nally, developers can provide users with uniform functionality regardless of message type, e.g. the ability to file send a picture to an RSS feed or over IM.

Whereas, HCI researchers are constrained to understand user behaviors (e.g. task management) within a particular messaging paradigm, e.g., e-mail, they can now begin to understand these issues, unfettered by such artificial restrictions. Furthermore, similar to developers, a unified approach to messaging via a unified data model gives HCI researchers the expressive capability they need to capture user level behavior metaphors that can support appropriate UI interactions.

Finally, users gain the most from our approach to messaging. Our prototype built on Haystack represents an evolutionary step in terms of providing a single, unified interface for supporting interpersonal and group communication. This unification produces the desirable effect of maintaining context and making information conveniently accessible by co-locating semantically related information, which is crucial from a usability perspective [4]. Such a capability has heretofore been impossible due to the segregated nature of existing messaging paradigms and their respective clients. By supporting a heterogeneous set of message types we have preserved and tied together the existing communication capabilities supported by popular systems such as e-mail and IM, resulting in a sum that is greater than its parts.

Some might argue that the plethora of messaging protocols is an advantage that should not be abandoned—that the existence of IM implies that a user's mailbox does not get filled up with the time-sensitive or off-hand comments that are common in IM, or that the existence of a news reader means that the user can decide to be "checking news" for some period of time without getting sidetracked by work-related e-mail. But, the unified approach sacrifices none of these benefits. Whatever "physical'" partition of the information is created by multiple protocols, can be simulated by a "logical" or "virtual" partition of information in the unified environment. Messages delivered by IM can be semantically marked so, and a view can be defined that shows only messages *not* delivered by IM. A specific folder can be defined that contains all arriving news. At the same time, the unified approach lets the user break down those partitions when it makes sense. For example, an instant message that triggered a lengthy e-mail discussion can be moved into the same conversation, while e-mail from certain mailing lists might want to interleave with the "news'" from RSS feeds instead of occupying the inbox because they both discuss the same topic.

We can leverage our data model to reduce the usability problems with current messaging systems. 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 [4]. This drawback is primarily due to a combination of the high rate of information arrival, users lacking time or being unwilling to file items away 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. By casting messages and conversations into RDF, we gain a persistent description to which we can add additional metadata that will improve searches and other information retrieval techniques. For example, a message can be annotated such that the user can associate other keywords with it such that future searches on those keywords will bring up the message. Hence, users will be better able to reduce "noise" and manage information overload by being willing to file information without worrying about not being able to find it later. By creating higher-level organizational concepts such as conversations and categories, users are able to consolidate different types of messages with similar topics together, reducing the clutter in users' inboxes, while having more intuitive ways to navigate through their message corpora. Furthermore, they can do this organization as needed, after the fact.

We have discussed various advantages of our system from the message receiver's perspective. However, there are some possible drawbacks for both the sender and receiver. With current systems, the sender does not have to decide how (which messaging system to use) to send the message. That decision is implicit in the application chosen to send the message. In our system, the user would explicitly need to specify where and how the message is being sent. However, we feel this should not be any greater a burden than it currently is; just a different kind of decision. Also, receivers who have so far relied on different application contexts to implicitly file their messages may now have to learn to file in the new system. Nevertheless, this can easily be overcome, by providing pre-packaged virtual folders that correspond to current application boundaries, e.g. e-mail messages, news, etc.

Finally, our RDF approach bodes well for unified messaging n the future Semantic Web and in fact helps facilitate it. We discuss this further in our Future Work section.

6. FUTURE WORK

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 to the status quo.

Finally, we hope to further exploit RDF's capability to encode arbitrary metadata in order to realize other usability benefits. By employing richer forms of semantic markup in order to better describe the nature of messages being sent, CSCW applications (e.g., workflow management) and automated filtering tools can be integrated into users' messaging clients. 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 information (e.g., negotiate meeting time and place). 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. Finally, senders can also begin to realize benefits by sending semantic messages such as a voting survey message, and have the responses automatically tallied, rather than have to open each message separately to get the results.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] Quan, D., Huynh, D., and Karger, D. "Haystack: A Platform for Authoring End User Semantic Web Applications." Proceedings of the Twelfth World Wide Web Conference 2003 (WWW2003).

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

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

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

[5] Nardi, B., Whittaker, S., and Bradner, E. “Interaction and Outeraction: Instant Messaging in Action.” CSCW ‘00.

[6] Whittaker, S., Terveen, L., Hill, W., and Cherny, L. “The Dynamics of Mass Interaction.” CSCW 98.

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

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

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

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

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

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

[13] Huynh, D., Quan, D., and Karger, D. “User Interaction Experience for Semantic Web Information.” Proceedings of the Twelfth World Wide Web Conference 2003 (WWW2003).

[14] Bellotti, V., Ducheneaut, N., Howard, M., Smith, I. “Taking Email to Task: The Design and Evaluation of a Task Management Centered Email Tool.” CHI 2003.

[15] www.kubisoftware.com

[16] Isaacs, E., Walendowski, A., Whittaker, S., Schiano, D.J., Kamm, C., “The Character, Functions and Styles of Instant Messaging in the Workplace.” CSCW 2002.

[17] Rohall S.L., Gruen, D., “ReMail: A Reinvented Email Prototype.” IBM Technical Report, TR2002-13.

[18] Handel, M., Herbsleb, J.D., “What is Chat doing in the Worklace?” CSCW 2002.

[19] http://www.microsoft.com/office/outlook/prodinfo/overview.mspx

[20] Werner, P., Liefield, T., Gilman, B., Bacon, S., and Apgar, J. URN Namespace for Life Science Identifiers. http://www.i3c.org/workgroups/technical_architecture/resources/lsid/docs/LISDSyntax9-20-02.htm

[21] Trastour, D., Bartolini, C., and Priest, C. Semantic Web Support for the Business-to-Business e-commerce lifecycle. Proceedings of WWW 2002.

[22] Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., et al. SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation. Proceedings of WWW 2003.