RDF Authoring Environments for End Users

Dennis Quan
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139 USA
dquan@ai.mit.edu

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

David F. Huynh
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139 USA
dfhuynh@ai.mit.edu

Abstract

The Semantic Web enables powerful agent-facilitated negotiation and retrieval functionality by enabling the transfer of machine-readable metadata across the Internet. This next generation Web presupposes the mass availability of metadata such as clothing measurements, busy/free time calendar indications, and dietary restrictions, meaning that users will need to provide more information to their systems to reap the aforementioned benefits. Current tools for capturing ontology-encoded metadata from users are ill-suited for this task, requiring knowledge of ontologies or the ability to navigate generalized abstract directed graphs. In this paper we present user interface paradigms based on the idea that an object can be represented on the screen by an extensible family of views. We present several strategies for allowing users to create RDF metadata by manipulating views, including a drag and drop form-based property editor and a view-based graph editor. Users benefit from being able to interact with objects by means of views suited to the context at hand, such as photograph representations or human-readable summary descriptions, rather than text fields with plaintext strings or URIs. Finally, we discuss a strategy for enabling advanced users to construct ontologies and customized views and to distribute them to users unfamiliar with modeling knowledge, ultimately giving end users intuitive interfaces for entering metadata.

Motivation

The Semantic Web provides a foundation upon which machines will be able to perform sophisticated coordination activities automatically, such as scheduling appointments, finding products that match specific criteria, and sharing ratings and opinions of goods and services over the Internet [3]. These activities require that specific pieces of knowledge, ranging from people's calendars to product specifications, be made available in machine-readable form. The Resource Description Framework (RDF) was designed specifically as a standard means of encoding such information for the Semantic Web [2]. As the amount of information recorded in RDF grows, the activities described above will start to become possible.

Technologies for exposing database information as XML or RDF have been developed and allow existing systems to participate in the growth of the Semantic Web. However, an important bottleneck to the proliferation of RDF as a platform for conducting everyday activities, such as coordinating schedules and sharing opinions, is the dearth of tools designed to allow end users to express information in RDF. Today's RDF authoring tools generally come in four flavors: (1) ontology editors such as Protégé [6], OilEd [9] and Ont-o-mat [4]; (2) graph-based representation viewers such as IsaViz [5]; (3) schema-specific user interfaces (the type automatically generated by database applications such as Microsoft Access and FileMaker) including Reggie [10] and Ont-o-mat; and (4) taxonomy editors exposed by tools such as Protégé and used by services such as the Open Directory Project [11]. However, existing tool implementations have for the most part focused on maintaining ontological constraints and not on addressing the HCI issues of information collection.

At the same time, a lot of the information that needs to eventually be provided in RDF is already being collected relatively successfully by current software. In fact, end user software employs many of the same approaches to information collection as those used by RDF authoring tools but is "streamlined" enough to be intuitive to non-technical people. For example, users know how to drag and drop contacts into the "To" field of an e-mail editor to fill in a message header form (header forms can be presented by many ontology editors). A simple row of radio buttons allows Web sites that sell books to collect rating information from customers. Diagramming software such as Microsoft Visio allows users to draw organization charts and relationship diagrams. Although not often considered as such, the kinds of information collected by the above user interface paradigms are also the kinds of information one wants to record in RDF for machine consumption.

Of course the interfaces associated with these examples are built from ontological constraints, such as the set of possible ratings for a book or the presence of a "To" property in an e-mail message header. Some users will understand how to put together basic schemas given the right tools, and the system should allow these users to conveniently share their schemas with others in the same community spirit as the current Web. Users can easily become familiar with incorporating Web content such as clip art, news articles, or tables into their documents and e-mails because of the ease with which Web browsers and word processors (through copy and paste or drag and drop) allow users to take advantage of publicly-posted information. Semantic Web client-side software must be able to allow users to take advantage of schemas posted on the Web just as easily.

Approach

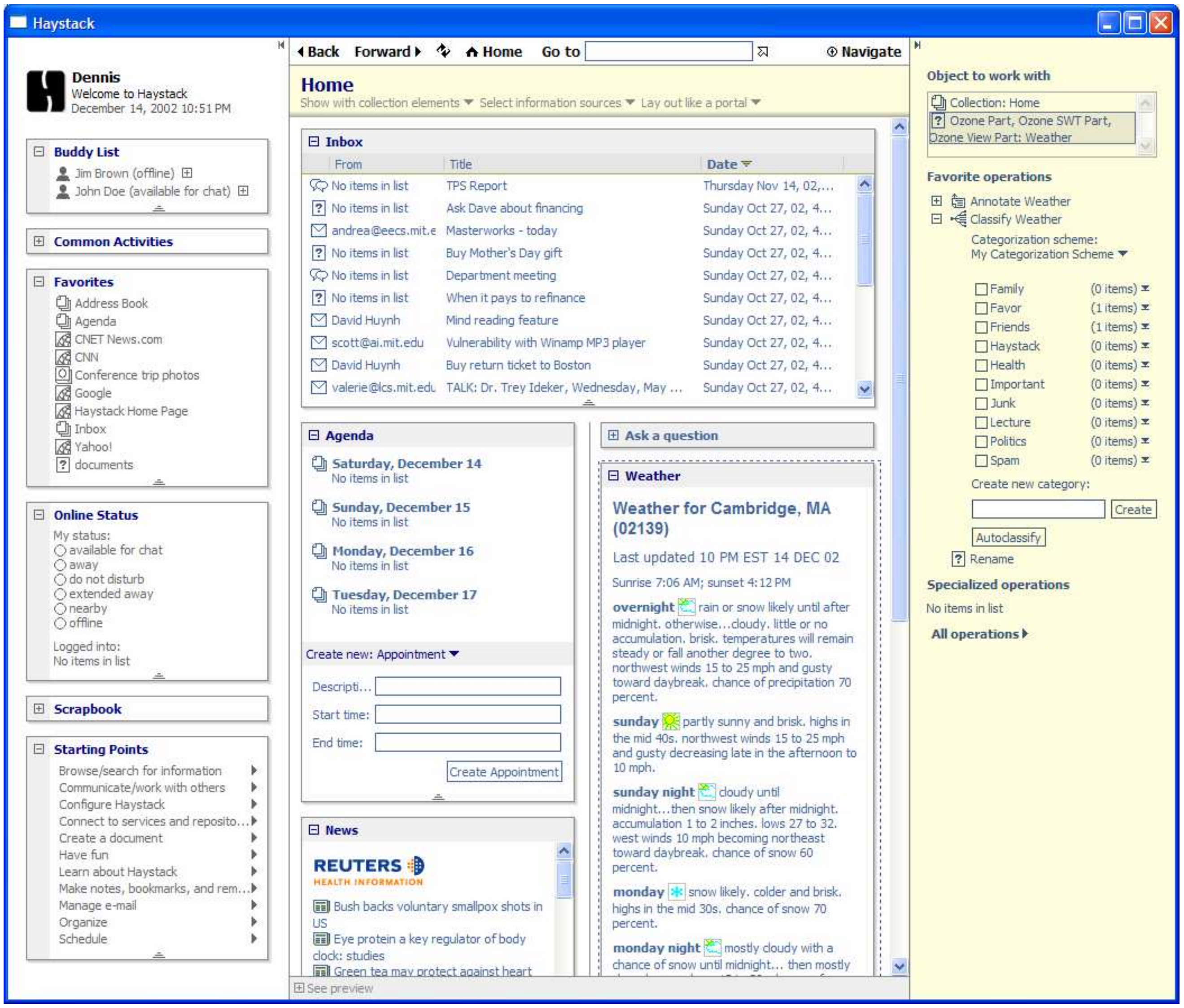
In this paper we present user interface paradigms for allowing users to record many different forms of metadata, i.e., create and manage properties of resources (the Semantic Web term for object) and relationships between resources.

The basic principle behind this paradigm is the idea of *views*—user interface components that serve as proxies for resources on the screen. Several views may be associated with any one resource; for example, a person may be displayed as an icon (an icon view) or as a large key-value pair listing as is the case in an address book (a property listing view). New views may be introduced into the system, allowing limitless freedom and flexibility in terms of how an object may be presented in a manner most suitable to the current context.

Exposing resources as views enables a number of advantages. One benefit is that users are made able to work with their information by *direct manipulation*, an HCI concept that has found to be successful in past research [7]. For example, users can drag and drop a view into a list as a means of indicating to the system that the resource represented by the view should be added to that list. Another advantage is that the problem of exposing a user interface by which users can manipulate the properties of a resource can then be cast as a problem of designing appropriate views for that resource, meaning that one does not need to rely on a handful of views to handle all possible kinds and types of resources.

Instead, we have developed several fundamental paradigms that correspond to various modalities in which users can specify information. These paradigms correspond to the four types of tools mentioned above. The most basic of these is the property editor, which allows users both to specify metadata and to customize the ontology by filling in or rearranging a form (types 1 and 3). We also describe paradigms for managing lists of resources and for placing resources into taxonomies (type 4). Finally, we present a graph editing paradigm that allows users to specify the relationships between resources visually when appropriate (type 2).

While these approaches for metadata input have been explored previously, what distinguishes the work presented here is the fact that views that embody the paradigms given here usually *embed* views of other resources. A property



Dennis
Welcome to Haystack
December 14, 2002 10:51 PM

Buddy List
Jim Brown (offline)
John Doe (available for chat)

Common Activities

Favorites
Address Book
Agenda
CNET News.com
CNN
Conference trip photos
Google
Haystack Home Page
Inbox
Yahoo!
documents

Online Status
My status:
○ available for chat
○ away
○ do not disturb
○ extended away
○ nearby
○ offline
Logged into:
No items in list

Scrapbook

Starting Points
Browse/search for information
Communicate/work with others
Configure Haystack
Connect to services and reposito…
Create a document
Have fun
Learn about Haystack
Make notes, bookmarks, and rem…
Manage e-mail
Organize
Schedule

Home
Show with collection elements ▾ Select information sources ▾ Lay out like a portal ▾

Inbox

From	Title	Date
No items in list	TPS Report	Thursday Nov 14, 02,…
No items in list	Ask Dave about financing	Sunday Oct 27, 02, 4…
andrea@eecs.mit.e	Masterworks - today	Sunday Oct 27, 02, 4…
No items in list	Buy Mother's Day gift	Sunday Oct 27, 02, 4…
No items in list	Department meeting	Sunday Oct 27, 02, 4…
No items in list	When it pays to refinance	Sunday Oct 27, 02, 4…
David Huynh	Mind reading feature	Sunday Oct 27, 02, 4…
scott@ai.mit.edu	Vulnerability with Winamp MP3 player	Sunday Oct 27, 02, 4…
David Huynh	Buy return ticket to Boston	Sunday Oct 27, 02, 4…
valerie@lcs.mit.edu	TALK: Dr. Trey Ideker, Wednesday, May …	Sunday Oct 27, 02, 4…

Agenda
Saturday, December 14
No items in list
Sunday, December 15
No items in list
Monday, December 16
No items in list
Tuesday, December 17
No items in list
Create new: Appointment ▾
Descripti…:
Start time:
End time:
Create Appointment

News
REUTERS
HEALTH INFORMATION
Bush backs voluntary smallpox shots in US
Eye protein a key regulator of body clock: studies
Green tea may protect against heart

See preview

Ask a question

Weather
Weather for Cambridge, MA (02139)
Last updated 10 PM EST 14 DEC 02
Sunrise 7:06 AM; sunset 4:12 PM
overnight rain or snow likely until after midnight. otherwise…cloudy. little or no accumulation. brisk. temperatures will remain steady or fall another degree to two. northwest winds 15 to 25 mph and gusty toward daybreak. chance of precipitation 70 percent.
sunday partly sunny and brisk. highs in the mid 40s. northwest winds 15 to 25 mph and gusty decreasing late in the afternoon to 10 mph.
sunday night cloudy until midnight…then snow likely after midnight. accumulation 1 to 2 inches. lows 27 to 32. west winds 10 mph becoming northeast toward daybreak. chance of snow 60 percent.
monday snow likely. colder and brisk. highs in the mid 30s. chance of snow 70 percent.
monday night mostly cloudy with a chance of snow until midnight… then mostly

Object to work with
Collection: Home
Ozone Part, Ozone SWT Part, Ozone View Part: Weather

Favorite operations
Annotate Weather
Classify Weather
Categorization scheme:
My Categorization Scheme ▾

☐ Family	(0 items)
☐ Favor	(1 items)
☐ Friends	(1 items)
☐ Haystack	(0 items)
☐ Health	(0 items)
☐ Important	(0 items)
☐ Junk	(0 items)
☐ Lecture	(0 items)
☐ Politics	(0 items)
☐ Spam	(0 items)

Create new category:
Create
Autoclassify
Rename

Specialized operations
No items in list
All operations

Figure 1: Haystack screenshot

editor uses views to represent values of properties rather than ordinary text field widgets, resulting in greater flexibility in terms of how complex properties can be presented. A graph editor also uses views to represent nodes in a graph instead of static, un-manipulable ovals labeled with URIs. Users benefit from interacting with their information in terms of graphical representations that are most appropriate to the context at hand (e.g., using a photograph view of a person instead of a URI for representing people in an organization chart).

The user interface paradigm presented here is embodied in our information management tool called Haystack [1]. Haystack is designed to help users easily manage their documents, e-mail messages, appointments, tasks, and other information. RDF forms the basis of Haystack's data model and is used to describe documents' properties and the connections between documents. Haystack's user interface, depicted in Figure 1, is composed of an extensive collection of views representing resources such as the user's inbox, calendar, favorites collection, and news reports. Haystack views are implemented in a combination of Java and an RDF scripting language called Adenine [1]. By taking advantage of the various paradigms described in this paper, Haystack enables users to describe RDF metadata such as e-mail headers, appointment details, document taxonomies, and even descriptions based on custom schemas such as flight itineraries and bibliographic entries. This metadata is then usable by Semantic Web agents and other programs that understand RDF.

View Architecture

At the heart of the paradigms presented in this paper is a user interface architecture specifically suited to presenting information in terms of views. Specifically, a view is a component that displays certain types of resources in a particular way. A given RDF class may have any number of different views associated with it. Furthermore, views are described in RDF, allowing a view to be characterized according to the RDF classes it supports and by the way it displays resources (e.g., full screen, in a one line summary, as an applet, etc.). When a resource needs to be displayed in Haystack in a certain way, such as full screen, a view is chosen that possesses the necessary characteristics.

As components, views enable pieces of user interface functionality to be reused. The developer of a one line summary view for contacts (perhaps displaying a person's name and telephone number) provides an RDF description to the system that enables developers that need to display summaries of contacts to reuse the component. The best example of reuse can be seen in the case of views that embed views of other resources. For example, a view of an address book containing contacts and mailing lists needs not implement views for displaying contacts and mailing lists; the system provides a way for views to specify that a resource needs to be displayed at a certain location on the screen in a certain fashion (e.g., as a one line summary). In this way composite views can be constructed that leverage the specialized user interface functionality of the child views that are embedded.

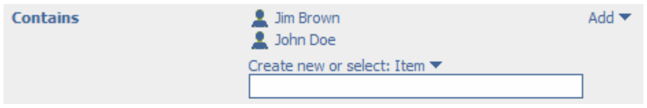
When a view is instantiated, the system passes the view a *context object* that informs it of the resource to be displayed. The context object also contains a pointer to the parent view's context object, if one exists as a result of a view being embedded within another view. In this way views are made aware of the context in which they are displaying information. For example, if an address book view is displaying a list of people by embedding individual person views, the person view can know not to display the "Add to Address Book" button, since it knows that it is embedded within the address book's view and hence is displaying a resource that is already in the address book.

Also, because the system is responsible for instantiating views and keeping track of where child views are to be embedded within parent views, the system can provide default implementations of certain direct manipulation features for free. A good example is drag and drop: When the user starts to drag on a view, the system knows what resource is being represented by that view, such that when the view is dropped elsewhere in the user interface, the drop target can be informed of what resource was involved instead of simply the textual or graphical content of the particular representation that was dragged.

Take the example of filling in a list of meeting attendees on a form. Instead of retyping or copying and pasting names of people from an address book, a user can drag and drop contacts from an address book into the list. Because the views representing contacts in the address book are associated with the resources they represent and not just the names of the contacts, the identities of the contacts' resources can be preserved. The alternative opens the possibility for ambiguity because information is lost. For example, what if there are two people named "John Doe" known to the system? Specifying the text string alone is not sufficient to disambiguate which John Doe is intended, even though it is clear that the John Doe desired is the one that the user selected in the address book.

Property Editors

The remainder of the paper presents various paradigms that result naturally from the basic concept of views serving as representations for resources. The first paradigm we present is the property editor, which is in essence a form in which users can edit the property-value pairs associated with a resource. A property editor is actually composed of a sequence of views of a specific type that display RDF properties such as "title", "creator", etc. The list of properties can be manually specified or derived from the RDF Schema definitions of the types of the edited resource. In addition to displaying the name of the property, an RDF property view detects from context what resource is being edited and displays a list of the values under the property by embedding views for each value, as depicted in Figure 2. (Literal values have a special view associated with them that enable literals to be edited as text strings.) Because property editors are simply lists of RDF property views, associating new properties with a resource can be accomplished by dragging and dropping properties into the blank area at the bottom of a property editor.



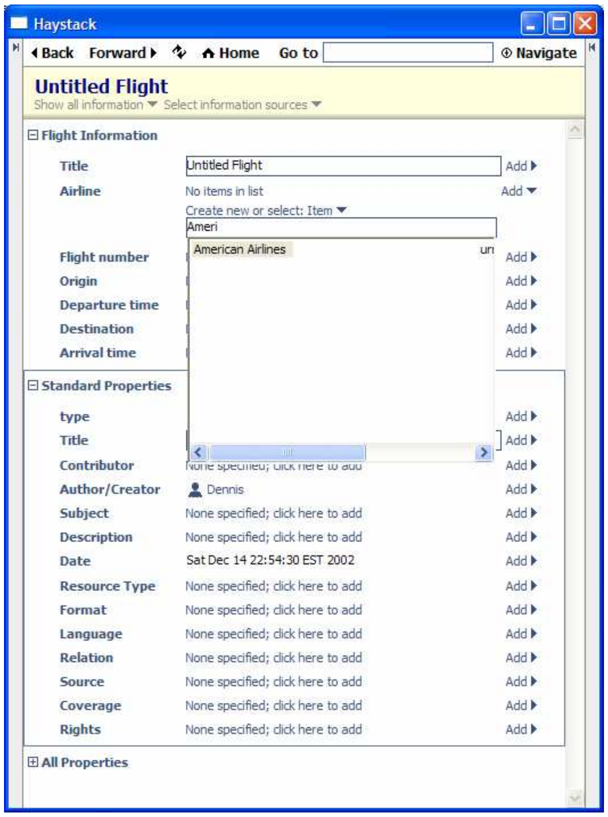
Contains	Jim Brown John Doe Create new or select: Item	Add

Figure 2: RDF property view of "Contains" property

This use of view embedding is in contrast to what most ontology editing environments available today provide, in which users must work with plaintext representations of properties and their respective values. URIs are meant to be computer-usable names for maintaining the identities of distinct resources in an RDF representation. While some URLs are easy to remember because of marketing (e.g., “http://www.priceline.com/”), the majority are not (in particular randomly generated URNs), and users should not be required to remember them or enter them into forms. In fact, we argue that users should not even need to see them, because users derive no useful benefit by seeing them. By using views to represent resources in the property editor and elsewhere, the system allows users to deal with familiar representations of resources such as icons and human-readable names.

This last point can be further emphasized if one considers that some resources serve as anonymous nodes for gluing together multiple parameters in an n-ary relationship. Take the example of a contact editor that exposes both a home phone number and a work phone number property. Suppose further that the range of these phone number properties is a phone number resource, encapsulating the country code, city code, area code, and local exchange number as properties. A user is not likely to consider a phone number a separate entity, one in which the separate properties of the phone number must be individually entered and displayed every time a contact is shown. In our paradigm, a specialized view can be provided to package together the various properties of a phone number resource and present them in a unified form. In other words, views can be employed to expose only the RDF property relationships important to the user and not the ones that are present in the data representation for structural and ontological reasons.

Despite our advocacy for a view-based approach, it is worth pointing out that there are times in which a textual specification is the most natural means of input, such as when a portion of the name of the resource in question can be conveniently recalled. We advocate the use of type-ahead support in these cases, such as that found in most modern integrated development environments, whereby the user would only need to type in as much of the name or description of the resource being described to uniquely match the text input with an existing resource in the system. Type-ahead support has been incorporated in the RDF property view in Haystack and is shown in Figure 3.



Untitled Flight

Show all information Select information sources

Flight Information

Title	Untitled Flight	Add
Airline	No items in list Create new or select: Item Ameri American Airlines	Add
Flight number		Add
Origin		Add
Departure time		Add
Destination		Add
Arrival time		Add

Standard Properties

type		Add
Title		Add
Contributor	None specified; click here to add	Add
Author/Creator	Dennis	Add
Subject	None specified; click here to add	Add
Description	None specified; click here to add	Add
Date	Sat Dec 14 22:54:30 EST 2002	Add
Resource Type	None specified; click here to add	Add
Format	None specified; click here to add	Add
Language	None specified; click here to add	Add
Relation	None specified; click here to add	Add
Source	None specified; click here to add	Add
Coverage	None specified; click here to add	Add
Rights	None specified; click here to add	Add

All Properties

Figure 3: Property editor with type-ahead window shown

Collections and Taxonomies

Separate from managing the properties of individual resources, another important use for creating RDF metadata is to organize resources into groups that share some set of characteristics. Two good examples of these are RSS feeds (collections of news articles published in tandem) and the Open Directory taxonomy. (Here we define taxonomy to mean a large hierarchical organization scheme in which documents or other resources may be classified at every level of the hierarchy, such as those exposed by the Open Directory or Yahoo!.) In this section we will examine paradigms for enabling users to create collections and taxonomies in RDF.

Taxonomy, perhaps one of the simplest forms of ontology, gives users what is in effect a series of unary predicates, which indicate an object’s membership in a collection. It is also one of the most highly employed forms of ontology; management of file system directory trees and Web page bookmarks are all examples. Finally, in addition to helping people organize files, classification into a taxonomy has the end goal of helping users find their information later [8]—a key motivation for maintaining metadata in Haystack.

Several views exist in Haystack for managing collections of resources. Fundamentally, each serves to embed views of a certain kind to display the resources that are members of the collection. Many different presentation styles result from this simple principle. For example, a collection view that embeds thumbnail views of its members can be used to display photo galleries (see Figure 4), while a view that embeds small card views can be used to display address books. Adding items to a collection can be performed by simply dragging and dropping items into the collection's view.

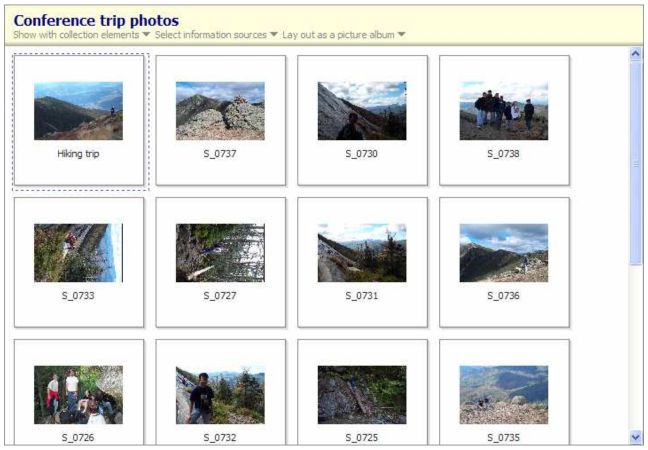


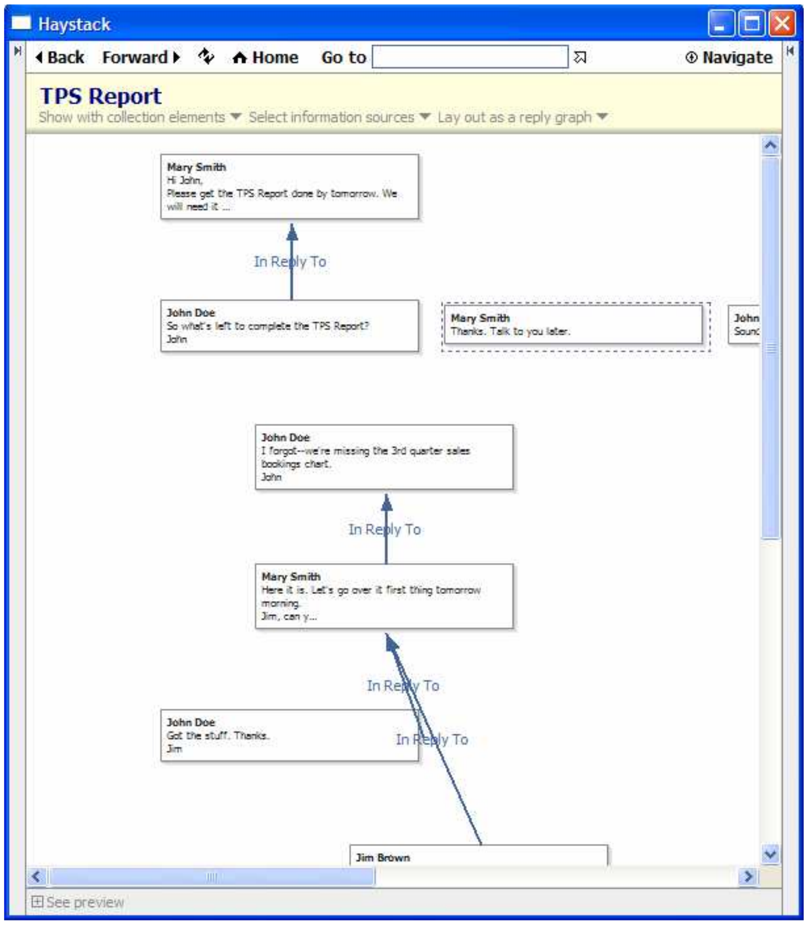
Figure 4: Example collection view with embedded thumbnail views

Furthermore, taxonomies—in essence, collections of collections—play a special role in Haystack. Users are given quick access to taxonomies for use in organizing their documents by means of the categorization pane, seen in Figure 6. This pane can be kept open while documents, Web pages, e-mails, or other objects are being viewed or edited. Placing an object into one or more collections becomes a simple matter of checking the boxes corresponding to the collections desired. (Currently, only one-level taxonomies are supported in Haystack; support is planned for deeper hierarchical taxonomies in the future.)

Graph Editors

Finally, we examine the problem of inputting metadata whose primary purpose is to express relationships. We postulate that manipulating directed graphs can be a natural paradigm for dealing with connections between resources because of its similarity to the means employed on paper for displaying connections between concepts—diagrams and charts—which can be especially effective means for recording ideas. Venn diagrams, UML diagrams, family trees, organizational charts, and process, causality or flow charts are used by people in a variety of fields to legibly express relationships between a set of entities or activities. Even looking beyond paper, we note that designers in a number of domains often use sticky notes and a whiteboard to conduct brainstorming sessions. Ideas are first recorded on the square sheets, attached to the drawing surface, and annotated with arrows or other connectors. In a similar fashion, grammar school students working on school projects often use note cards to capture ideas from various sources, and then lay out the cards in order to solidify the organization of their final reports.

Because diagrams are such a powerful means for displaying and capturing information, we consider graph editing to be a key paradigm for interacting with RDF data when the user wishes to focus on the relationships between resources. A graph editor can present a collection of RDF statements in the obvious manner by representing resources as nodes and properties as arcs connecting nodes. However, taken to its extreme, graphical representations can be extremely misleading to users. Although from an ontological perspective, we can model the relationship expressed in the sentence "Bob is 25 years old" by two nodes named "Bob" and "25" connected by an arrow labeled "age", this notion may not be intuitive to users, who are used to age being a property of people that is entered in a form-like fashion. Even for data that users usually regard as relationship-oriented, one must be careful to only display the relationships and nodes that are important at any given time.



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
Sound

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

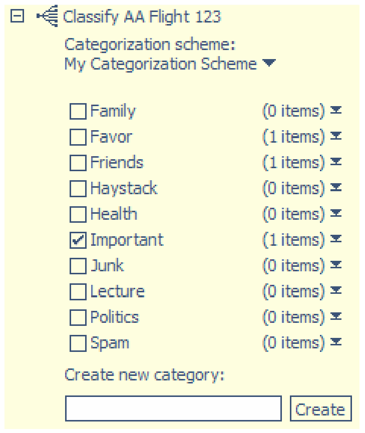
Mary Smith
Here it is. Let's go over it first thing tomorrow morning.
Jim, can y...

John Doe
Got the stuff. Thanks.
Jim

Jim Brown

Figure 5: Use of a graph editor for displaying a reply graph

To address these issues, graph editors can embed different types of views to display the resources being manipulated in order to control the level of detail being presented. Figure 5 shows an example of Haystack displaying a conversation as a graph of messages. Although the ontology for messages includes properties such as “From”, “To”, and “Body”, these fields are not visualized as arcs. Instead, the type of view that is specified by the conversation’s view for embedding displays a snippet of the message’s body as well as an indication of who sent the message. The focus is placed on the “in reply to” connections that exist between messages. To a user looking to gain an idea of the “big picture” of the flow of the conversation, the approach adopted here is arguably more useful than one in which all RDF resources present in the data representation of this conversation, such as the originating and destination e-mail addresses in the “To”, “CC”, and “BCC” fields, the message



Classify AA Flight 123

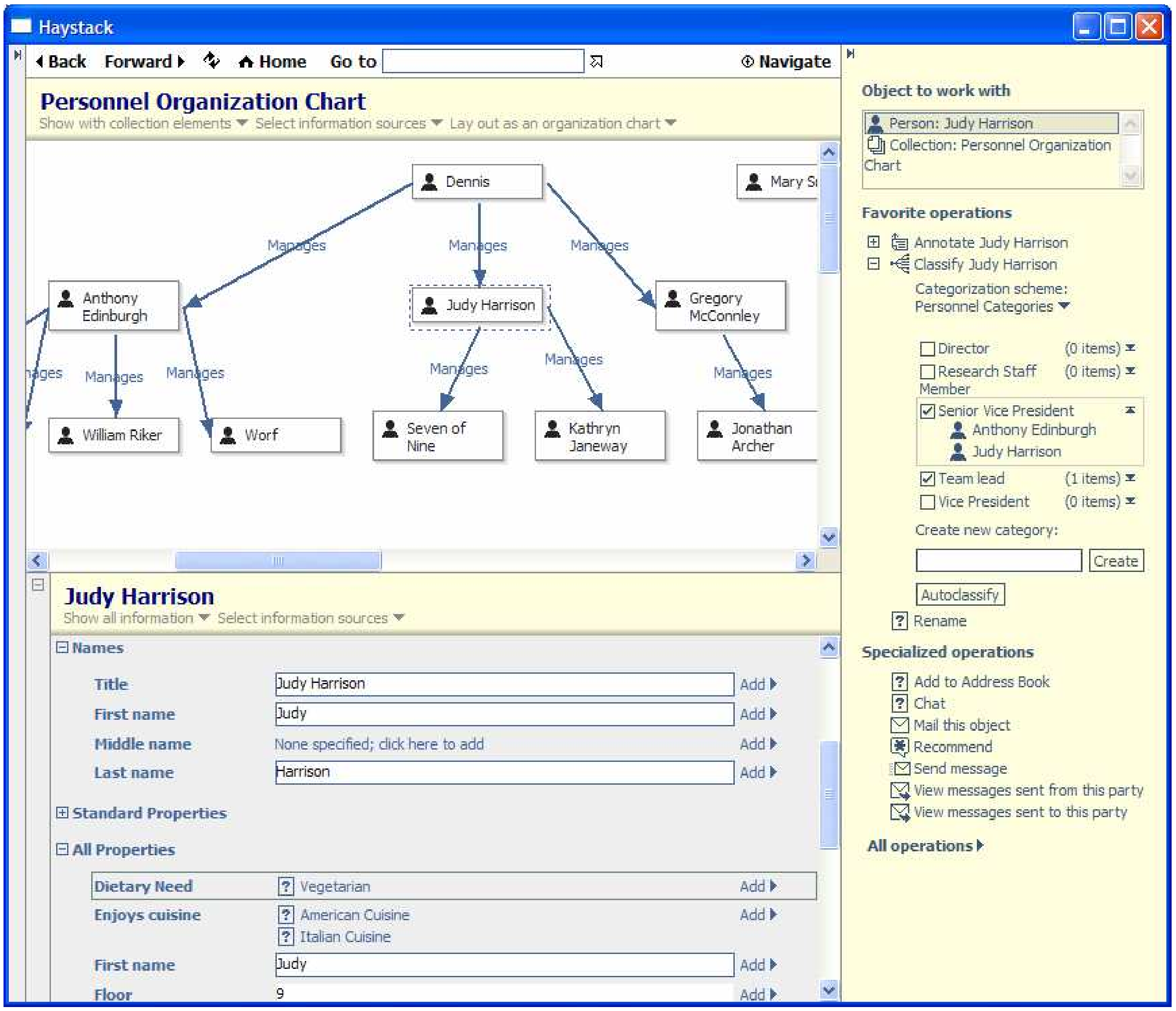
Categorization scheme:
My Categorization Scheme

☐ Family	(0 items)
☐ Favor	(1 items)
☐ Friends	(1 items)
☐ Haystack	(0 items)
☐ Health	(0 items)
☑ Important	(1 items)
☐ Junk	(0 items)
☐ Lecture	(0 items)
☐ Politics	(0 items)
☐ Spam	(0 items)

Create new category:

Create

Figure 6: Categorization pane



Haystack

Back Forward Home Go to Navigate

Personnel Organization Chart

Show with collection elements ▾ Select information sources ▾ Lay out as an organization chart ▾

Judy Harrison

Show all information ▾ Select information sources ▾

⊟ Names

Title	Judy Harrison	Add ▸
First name	Judy	Add ▸
Middle name	None specified; click here to add	Add ▸
Last name	Harrison	Add ▸

⊞ Standard Properties

⊟ All Properties

Dietary Need	Vegetarian	Add ▸
Enjoys cuisine	American Cuisine Italian Cuisine	Add ▸
First name	Judy	Add ▸
Floor	9	Add ▸

Object to work with

Person: Judy Harrison
Collection: Personnel Organization Chart

Favorite operations

⊞ Annotate Judy Harrison
⊟ Classify Judy Harrison

Categorization scheme:
Personnel Categories ▾

☐ Director	(0 items)
☐ Research Staff Member	(0 items)
☑ Senior Vice President Anthony Edinburgh Judy Harrison	
☑ Team lead	(1 items)
☐ Vice President	(0 items)

Create new category:

Create

Autoclassify

Rename

Specialized operations

Add to Address Book
Chat
Mail this object
Recommend
Send message
View messages sent from this party
View messages sent to this party

All operations ▸

Figure 7: Example of paradigms working together

bodies, and the attachments (as well as the predicate links connecting them) are visualized. And, as before, using views to render resources to the screen allows the system to display arbitrary collections of resources without requiring graph editors to have any hard-coded notions of how to display graph nodes.

In addition to enabling users to visualize relationships, support is planned to allow users to select an RDF property/predicate from a list and to drag arcs from one node to another. This list can be derived from an ontology by selecting all properties whose domain and range types are those possessed by resources in the graph. The palette of possible arcs will be displayed as other collections are displayed—as a sequence of views—meaning that other predicates can be added to the palette by means of drag and drop. Similarly, resources (nodes) can be added to the graph by dragging views of those resources into the graph.

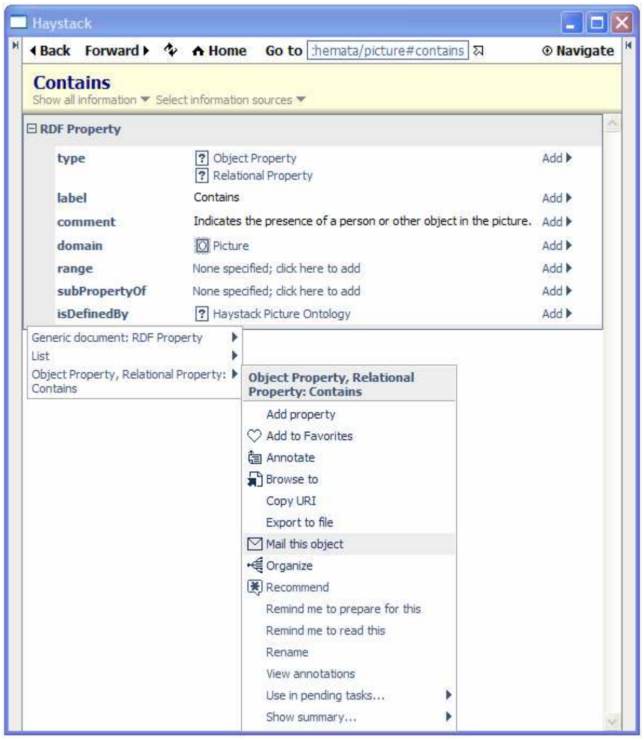
Example Scenario

We feel it is important to emphasize that the various paradigms presented here are complementary and can be used together to construct sophisticated user interfaces for working with various kinds of metadata. Figure 7 illustrates a screenshot from Haystack displaying an organization chart in the graph editor that allows a user to work with the relationships between various people. In addition, the preview pane located on the bottom portion of the figure depicts examples of the property editor, whereby arbitrary properties (as specified by an ontology) can be entered, such as dietary requirements or names. Finally, the pane on the right side of the screen shows the categorization pane and an embedded collection view showing who else is indicated to be a "Senior Vice President."

In theory it would be possible to present the entire display in either the property editor or the graph editor, because both editors are general enough to support the entire RDF data model. However, the figure illustrates that despite the fact that "enjoys cuisines" and "manages" are both RDF properties in the modeling sense, for the purposes of this interface the choice of which paradigm to use to display these properties is key to providing an intuitive user experience.

Future Work

In this paper we have discussed how views play a key role in giving users intuitive representations of resources present in RDF metadata as well as several user interface paradigms that utilize views to enable users to create RDF metadata based on ontological specifications. Together, views and ontologies provide users with methods of structuring the information they wish to input into their systems. However, when a user's system encounters new information that is written to a foreign



type	Object Property Relational Property	Add
label	Contains	Add
comment	Indicates the presence of a person or other object in the picture.	Add
domain	Picture	Add
range	None specified; click here to add	Add
subPropertyOf	None specified; click here to add	Add
isDefinedBy	Haystack Picture Ontology	Add

Figure 8: Context menu of an RDF property showing "Mail this object" command

ontology, the system will need a way to retrieve the ontology and any corresponding views. Similarly, when a user wishes to express a relationship or talk about a resource that is not known by the system, either the system must allow the user to coin a URI to represent the new concept, or the system must determine if others have already named this concept.

Looking towards the future, we are investigating the use of RDF-enabled search engines and shared repositories for helping users locate concepts, people, properties, or even views named in ontologies defined by others. Search en-

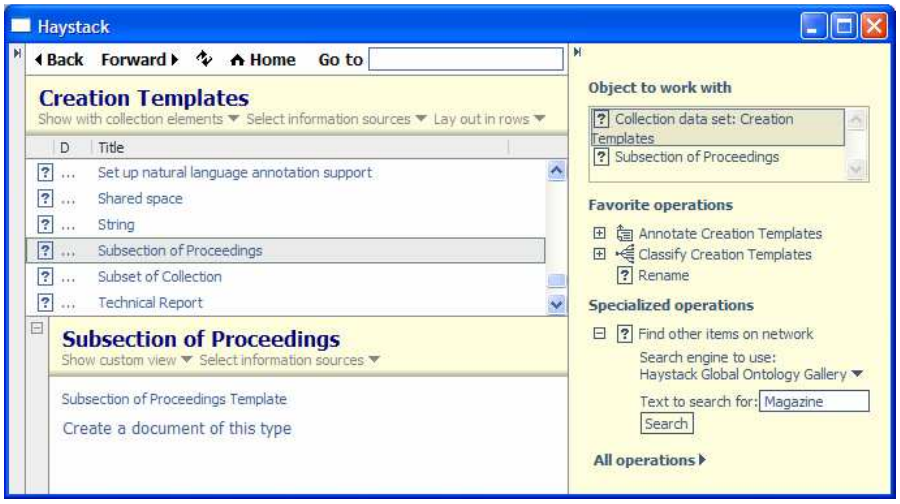

Haystack
Back
Forward
Home
Go to
Creation Templates
Show with collection elements
Select information sources
Lay out in rows
D
Title
Set up natural language annotation support
Shared space
String
Subsection of Proceedings
Subset of Collection
Technical Report
Subsection of Proceedings
Show custom view
Select information sources
Subsection of Proceedings Template
Create a document of this type
Object to work with
Collection data set: Creation Templates
Subsection of Proceedings
Favorite operations
Annotate Creation Templates
Classify Creation Templates
Rename
Specialized operations
Find other items on network
Search engine to use:
Haystack Global Ontology Gallery
Text to search for:
Magazine
Search
All operations

Figure 9: Screen from which user can select the type of resource to describe

gines would accept descriptions of what is being sought, such as a property whose label includes the text "favorite color" or a one line summary view for a hotel reservation, and perform an RDF graph match against the metadata available to it; in response a search engine might return information on how to contact another server on the Semantic Web to retrieve the actual information. A shared repository would actually store the relevant information and return the information directly to a requesting client. Views would be similarly resolvable under this scheme because view characteristics are described in RDF.

Support for RDF search engines would come in two parts. First, a user's authoring tool would need to expose convenient and intuitive mechanisms for connecting to search engines and for incorporating new ontologies and views. Figure 9 shows the screen in Haystack from which a user can select the type of resource to describe. If the user does not find the type needed, he or she can use the pane on the right side of the screen to invoke a query to find an applicable ontology. Haystack could then download the required metadata and incorporate it.

Second, ontology editors would need to make it easy for ontology designers to publish and publicize their ontologies and views. Haystack contains built-in support for electronic messaging, including an interface to POP3/SMTP e-mail and the ability to extract portions of an RDF graph relevant to a specific resource. Any resource in the system can be sent to others via e-mail by right-clicking on a view of the resource and selecting "Mail this object" from the context menu (see Figure 8). Similarly, customized ontologies or views could be e-mailed to search engines or shared repositories. We are currently constructing a shared repository that can accept such e-mails and expose querying functionality.

While the solution outlined above will allow users to reuse and share ontologies, it does not address the problem of multiple ontologies specifying models of the same domain. One possibility is to employ mappings or inference rules that would allow the user's environment to translate a resource's metadata into whatever ontology is required. These mappings or rules could be transported by means of RDF search engines or shared repositories in much the same fashion as views and ontologies as described.

Acknowledgements

This work was supported by the MIT-NTT collaboration, the MIT Oxygen project, and IBM. The authors wish to thank Jimmy Lin for his comments on drafts of this paper.

References

[1] Huynh, D., Karger, D., and Quan, D. Haystack: A Platform for Creating, Organizing and Visualizing Information Using RDF. *Semantic Web Workshop, The Eleventh World Wide Web Conference 2002 (WWW2002).*

[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] Handschuh, S., Staab, S., and Maedche, A. CREAM—creating relational metadata with a component-based ontology-driven annotation framework. *Proceedings of K-CAP 2001*, October 2001.

[5] Pietriga, E. IsaViz. http://www.w3.org/2001/11/IsaViz/.

[6] Eriksson, H., Fergerson, R., Shahar, Y., and Musen, M. Automatic Generation of Ontology Editors. *Proceedings of the 12th Banff Knowledge Acquisition Workshop*, 1999.

[7] Shneiderman, B. Direct manipulation for comprehensible, predictable and controllable user interfaces. *Proceedings of IUI 1997*.

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

[9] Bechhofer, S., Horrocks, I., Goble, C., Stevens, R. OilEd: a Reason-able Ontology Editor for the Semantic Web. *Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, Springer-Verlag LNAI*, vol. 2174, pages 396–408.

[10] Distributed Systems Technology Centre Resource Discovery Unit. Reggie—The Metadata Editor. http://metadata.net/dstc/.

[11] Open Directory Project. http://dmoz.org/.