Aaron Zinman

Personal Metadata: Order from Clutter

Cognitive Science 190 2002-2003: Honors Thesis

Advisors:
David Kirsh
John Batali
Dan Bauer

PURPOSE

How can we improve people's work by adding digital metadata to their environment? Metadata, construed broadly, is information about objects and their related events and processes. The most familiar forms of metadata are subject, title, author, date and related descriptions of articles, papers and books. But the concept is comprehensive enough to include annotations found on a document, data about how a document has been used, when, by, whom and how often, and even where it is to be found on a desk or in a library. My objective in this research project is to explore how to digitally augment work using AI techniques to track some of the significant interactions that occur in them.

PROBLEM

Metadata has long been used to augment physical environments. Libraries, in particular, need a method by which books and other artifacts can be efficiently tracked and identified by multiple attributes. Standards such as Open Geospatial Interoperability Standard (OGIS) have been developed to solve specific superficial problems: facilitating interoperability of data across sources. The World Wide Web attempted to move beyond basic level metadata. Tim Berners-Lee developed it with the vision of metadata linking together documents in an infinitely large semantic web. Yet there has been little attempt at recognizing and understanding the deeper structure of local environments: digital or physical.

Figure 1. Sketch of an office of the future using a hybrid environment.

Figure 1 is an artist's depiction of a future distributed hybrid environment in which team members in offices at UNC and CMU are relying on virtual objects to help them collaborate. Such research may be a forerunner of the future but it puts too little emphasis on the cognitive questions that arise when people use digital tools to augment their physical environments. For example, what kinds of mental representations of the environment and its contents are the collaborators sharing? Is their conceptualization of the same information identical? What factors influence the implicit understanding of these virtual objects? Do they affect all people in the same way? What deeper structures

underlie the interaction between the objects and the environment, and how can it be modeled? How can we use cognitive engineering to optimize and record the mental projection of metadata onto these objects? Can we store this metadata in a natural way to later facilitate better and easier retrieval?

People project structure onto their environment, saturating objects with meaning. The document I last touched, the article that X gave me, the paper I put all my annotations on. All these descriptions are associations and attributes that help people manage the resources around them. Often they are work related, task specific, personal, and context relative. The power of distributed cognition comes from the ability for people to communicate their projected structures, or metadata, to aid others. This communication may be explicitly given by the individual, or is implicit based on the situation, common beliefs, cultural understandings, etc. How can we harness these properties using digital support to help facilitate collaboration as well as understanding of our own environment? Can a system discover the effective description of the environment that allows machine inference of the function of office resources, which may not be obvious for the casual observer? This project aims to address all these issues in a way that facilitates an optimization of workflow by solving a small part of a much bigger system.

BACKGROUND

The current idea of this project comes from a shift of focus from the first iteration of the project design. The first iteration was to implement enhanced digital supports for a hybrid Physical-Virtual Collaborative Environment. A *Virtual Collaborative Environment (VCE)* is defined as (Churchland, 2001):

> A computer-based, distributed, virtual space or set of places. In such places, people can meet and interact with others, with agents or with virtual objects. VCEs might vary in their representational richness from 3D graphical spaces, 2.5D and 2D environments, to text-based environments. Access to VCEs is by no means limited to desktop devices, but might well include mobile or wearable devices, public kiosks, etc.

We had decided that a hybrid VCE was a natural evolution of the concept, allowing participants to use physical objects alongside virtual objects in a hybrid system. Using computer vision to track and identify previously known objects, digital supports such as Onoto pens and Tablet PCs to capture live annotations, and rear-projected tables to present virtual copies of physical documents that reflect live changes, the traditional modality of VCEs were extended significantly. While many researchers are currently looking for ways to create more natural setups for projecting the virtual world into the physical (Stuerzlinger, 1998; Fuchs, 1998), there is a lack of concentration on how the two worlds interact. We decided that the deepest problem of VCE's is not how to inhabit and organize actions in VCE's, but how to live in the physical world while working with digital elements.

It is very desirable to have the ability to track and recognize physical objects for Virtual Collaborative Environments for several reasons. First, the biggest problem with

remote collaboration is the lack of a distinctive shared physical space. Normally, agents interact with the ability to pass objects between each other and leave currently unused but relevant items on a shared physical table. It is very natural for the agents to be able to look freely directly at any person or item and process any stimuli accordingly. In typical video conferencing, these natural encounters can be difficult to reproduce. There is not sense of a shared common space, other than what some software allows for in terms of either a whiteboard or one particular document that both sides have agreed to edit in tandem. It is nearly impossible to search a remote location for a sense of what the other agent has to offer as it could in a physical conference room (i.e. shuffling papers). Implementing digital tracking and identification, a virtual space may be created that mimics the physical environment, allowing for free flow of information to be processed by any agent in a natural manner. With digital support, annotations and other metadata can be shared with the object.

Within our hybrid environment, we have objects and actors. Objects typically are physical or digital documents and have a central location of storage of their metadata. Actors are the collaborators in the environment and can perform various actions on the objects. The network tracks these actions and stores them in the objects metadata. However, it is not only the explicit actions that are stored in the metadata. Higher-level relations between objects and each other as well as objects and their environment play a key role in our system. When an actor puts a document on top of a stack of other papers, a relation is created between the adjacent papers on the stack and the new document. To describe relationships between objects in the environment using metadata is to create ontologies. Ontology is well defined by Broekstra et al. as:

> … a formal, explicit specification of a shared conceptualization. A *conceptualization* refers to an abstract model of some phenomenon in the world which identifies the relevant concepts of that phenomenon. *Explicit* means that the type of concepts used and the constraints on their use are explicitly defined. *Formal* refers to the fact that the ontology should be machine processible, i.e., the machine should be able to interpret the information provided unambiguously. *Shared* reflects the idea that an ontology captures consensual knowledge, that is, it is not restricted to some individual, but accepted by a group.

This project uses an ontology to be able to appropriately describe the environment and its transactions. This ontology provides a descriptive language for querying the environment using defined notions of meaning, including any appropriate thresholds or other qualifiers. Each rule in the ontology has been crafted by hand and is not learned.

It is common to find ontologies expressed in a common file format. One such popular XML-based language is RDF (Resource Description Framework) and it is now a W3C standard. It supports the ability to describe objects using a common vocabulary, allowing an agent to be able to understand any part of a definition that it knows about. This is typically used to share information across non-compatible databases. Since systems such as OGIS have been created, many proposals to generalize metadata brokerage have surfaced (Kashyap, 2000; Maedche, 2002). Many schemata have been designed to run as layers on top of RDF, using building off RDFS (Resource Description Framework Schema). One popular extension to RDFS is OIL (Fensel, 2001).

```
<rdfs:Class rdf:ID="herbivore">
      <rdf:type
rdf:resource=http://www.ontoknowledge.org/oil/RDF-
schema/#DefinedClass />
      <rdfs:subClassOf rdf:resource="#animal" />
      <rdfs:subClassOf>
            <oil:NOT>
                  <oil:hasOperand rdf:resource="#carnivore"/>
            </oil:NOT>
      </rdfs:subClassOf>
</rdfs:Class>
```

Figure 2: In this example Fensel provided, we have defined an herbivore to be an animal but not inherit the properties of a carnivore

OIL allows us to comprehensively tag explicit metadata in a simple fashion, providing a straightforward way to incorporate the ways that data can be used across all components of the system. Additionally, the model forces us to take an object-oriented approach, which reinforces good coding practice in an environment where object-orientation is the foundation. Furthermore, OIL allows us to define ontologies in ways such that the skeleton or framework of the ontology is readable by humans in a logical fashion, which both facilitates creating and understanding computer-generated ontologies. It is hoped that future versions of this project will use OIL as its knowledge base.

Recent work in visualizing GIS information has used organizational cognitive theories for navigating and classifying data effectively. Their methods primarily deal with the fact that "our stored knowledge is, generally speaking, organized by *kinds* vertically within hierarchy (taxonomies), and by *parts* (partonomies) within categories. At the basic level, our knowledge is mostly concerned with *parts*" (Mennis, 2000). They continue categorical theory to rely on attributes of specific objects as classifiers under a broader category. These attributes rely on a "*graded* internal structure" to a category to define its parts. For example, a "mug" might reflect a slight morphological change to the "cup" category. Mennis also builds on the Triad framework where "entities [are decomposed] into interrelated location-based (where), time-based (when), and object-based (what) information" (Mennis, 2000). The "what" of an object holds its relation, both to a super-structure (category) and its unique attributes (parts). The where and when

of an object can be tracked using video cameras and digital support. Utilizing these theories, we can build our metadata modeling cognitive storage for easier retrieval. If humans construct linguistic meaning from non-linguistic conceptual systems (Potter, 1986), then perhaps knowledge of a document spanning all its aspects is stored similarly. If this is true, then various attributes can be used to identify documents within a contextual setting. For example, given a combination between temporal relation ("you used the document before working on the Minendez Case") and specific parts ("it contains graphs, financial information, and is approximately 6 pages long"), one might be able to remember the specific document that is being queried. This natural method of query can be done to the system in the same way one might query another person, using only the parts of the Triad framework that they can remember.

One of the more significant aspects of the triad framework is the "what" category. People have to accommodate in their mental representations of objects ordinary metadata and personal metadata. Ordinary metadata are categories that the Dublin Core and MARC look at, such as subject, title, and author. In a physical world, ordinary metadata could consist of colors of a paper, dimensionality, and font size. Personal metadata is a unique representation of the same object due to the various relationships, contexts, and events that occur separately to each individual. The foundation of personal metadata is the events and activities that shape the types of interactions an object may have with its environment. These actions may determine the fate for one object or desire for another. Often ad-hoc categories are projected onto objects based on a number of factors. History of objects leads to strategies of how to manage items which may not be relevant to the immediate context. Whittaker and Hirschberg discovered strategies of piling and filing

that were independent to a person's job type. In their study, users effectively switched representations of known objects to re-categorize for future retrieval. Knowledge of the future is often projected by creating environments which aid to the retrieval of situated objects. Creating a pile for things that are needed in the meeting later today is one possible method. Typically much of the information needed to understand relations between ad hoc groups can be difficult to assess from the explicit data on the objects. It is simply not needed when personal metadata is able to efficiently use little queues to recall categorizations.

The uniqueness of personal metadata leads to great difficulty with identifying ad-hoc categories using AI. This is because "people conceptualize a category differently across situations, with each conceptualization embedded in a background situation. A single situation-independent concept does not represent the category; nor does the concept only represent the category in isolation, independently of the situations in which it occurs" (Barsalou, 2000). For example, "cheese, Frank, Madonna CD" could be items that belong to the categorization "things I am bringing to the picnic". Typically these items would never have been associated, and probably never should be. Generating and keeping semantic links between such objects from previous ad hoc relations has no real meaning outside the current context and is more likely to generate false positives on future searches. However, the situation "going to a picnic" is very different from the usage this system was intended to capture. In real-life working environments, *active* objects that are commonly in the same ad hoc categorization are likely to have meaningful semantic links. An object is said to be *active* when it is frequently accessed, modified, or moved relative to all of objects in ones environment. For example, when

working on a particular project certain textbooks or documents may be constantly piled together. The significant activity of these objects hints that their placement is shaped by much more thought than if they were placed so the user could shift focus.

However, it is unwise to make assumptions of thresholds for linkage based on activity. Most studies of ad hoc categories have demonstrated that people tend to judge items differently—a matter that becomes worse when compounded with different working strategies (Kurts, 2001). Knowing the statistical averages of piling, filing, and organizational strategies of the users is necessary for reliable automated semantic linking based on usage. And that's just for basic office maintenance.

METHOD

Overview

To address the issues of digital support using metadata, I am looking towards a "Big Brother" system that monitors your actions, interprets them, and then stores them for later retrieval.

Monitoring

Ideally the system would monitor a work environment in the physical and digital domains. In the physical world, cameras placed throughout the working environment, combined with RDIF tags and sensors would be aware of the locations and identities of

all objects and persons in the physical environment. Enhancers such as Onoto pens would allow annotations and any other writing on paper in the context aware office to be captured and noted by the system, categorized by what was the note was taken on. Originally, this project attempted to give a proof of concept Big Brother support for automatic document identification and tracking in a constrained environment, but in the interest of time the project shifted focus towards digital-only support.

In our digital world, events in an object-oriented graphical environment can be captured with much more ease than their physical counterparts. All objects that enter, get called by, or leave the system during sessions may be annotated by an event stream interpreter using higher level rules. For example, instead of simply noting that a particular program created a file some time after opening a previous one, the system sees the type of program (WinZip), notes the action taken in the program (decompress), and creates a meaningful link between the compressed file and its output. This way we can later trace the files to their original archive, along with any meaningful relationships the original archive may have. In this project, we have ignored the problem of monitoring events automatically. Interpreting system-level calls is not only operating system and application dependent, but is not the type of scientific problem we're attempting to solve. Problems of interpretation have been looked at and are solved using the ontology I have constructed in a system impendent way. Furthermore, the system should interface with existing environments and applications to allow additional metadata to be placed without the need to change existing representations.

Data is needed to inspire the design of an ontology as well as information to process to assess the system's reasoning. To acquire data I captured 120 minutes of

video of myself preparing for the Honors Presentation. The video was then annotated and put into a database according to my ontology.

Interpretation

In order to interpret the events that are recorded, as well and provide a consistent language for querying the database, an ontology is needed to provide definitions to define and represent the environment. Much of the intelligence of the system comes in its representation of events. To create the ontology I decomposed many tasks, separating out key parts to allow for a modular design. Additionally, I generated many queries that I wished for the system to be able to answer. For each question, I determined the plausibility for my system to be able to answer it, as well as what limitations the question should have. Using the pseudo-algorithm, I then modified my ontology to reflect the structure of the current search as well as the previous. This ontology is not intended to be the complete ontology for the envisioned system, but rather one that provides adequate description for the digital world. However, much of the ontology was created with flexibility in mind to be able to scale to the physical world.

Retrieval

Information is retrieved from the database using a combination of a reasoning engine as well as some form of a GUI manager. In the envisioned system the user would use some form of natural language querying and/or a 3D virtual world that would allow people to ask spatial queries more intuitively. In this project a rudimentary GUI was created, however it was not the primary focus and received much less time compared to the rest.

Motivation

The system must have an understanding of how to model the environment for relations to have valid meaning. Accurate "understanding" is achieved by human-designed ontological representations rather than machine learning. While much research is currently being put into ontology learning, typically environments which provide good results are heavily bounded (Maedche, 2002). This author feels that ontological learning as a basis for foundation of representing the user's environment is not only be inappropriate application for this project, but the current status is far enough away that it could easily shift the main focus away from the current goals. Unfortunately, there are future aspects of this project which may require learning for the individual user, yet the principal structure should remain the same.

States may change with the position of the individual nodes (although they are not the defining factor for a change in state), altering the context of the document. For example, a document's explicit metadata about its contents and author would not change as the document moves from a desk to inside a filing cabinet, but its implicit taxonomy

changes due to its new inherited classification. This is compatible with the Triad framework, where the documents category changes but not its parts. Because the system maintains a history of the states, relations due to past context may be preserved if a certain indexing heuristic requires previous associations. Furthermore, a heuristic may take advantage of the temporal relations between associations to other objects and their previous states. This aspect of the system is extremely useful for visualization.

Documents can contain explicit metadata that do not rely on any inherited property, but instead could be entered by the user. At any point, subject, keywords, author, related works, amongst other items can be independently stored metadata. Keywords are single references, and are not cross-linked to avoid excessively high dimensionality, which has been shown to give worse results in machine-generated categorization (Wang, 2002). Relations to abstract entities such as stacks of papers are stored differently depending on their stability. Stable category memberships could be links to known categories that are not run-time generated, or are ad-hoc categories for which some determining factor has granted elevation to permanent status. Examples of a stable category would be “Computers”, “Cognitive Science”, or “Recipes”. Some determining factor (ontological learning, bigram categorization, or manual input) would need to explicitly create the membership to a stable category. Unstable relationships, such as stacks that form ad-hoc categories, would remain nameless unless specifically told otherwise by the user. Additionally, their status and meaning changes as they are created and destroyed, increasing the complexity for effective searching.

Stacks of papers can be analyzed for their content. Each stack itself is an ad-hoc creation: a temporary grouping that contains one or more implicit relationships. The

relationships could be of any ontological combination, from papers that are related to a certain subject to things that were in my inbox at 10 AM. Other types of objects can be attached to each other, such as Post It notes and annotations. Post Its are different from articles, which are different form other types of annotations, giving each a different meaning and ontology/context. They are both objects yet have a strong link to other documents, influencing the heuristic for association more than any other type of object. Items such as annotations are stored in the model embedded in a document, because their existence only directly references one object.

The system should exhibit both modularity and mutuality. A system exhibits *Modularity* when "all elements of the system exist within certain contexts, in relation with other elements" (Ruhl, 1989). Conversely, a system exhibits *Mutuality* when "elements are not merely related, but mutually defining. The status of each element determines and is determined by other elements; any change in one element means a change in the nature of others" (Ruhl, 1989). In this system, each object modularly can exist on its own or within a specific framework, but the system benefits from its ability to create mutual links of abstract structures, enhancing later retrieval with new types of objects to search for.

Actualities

Ontology

In theory the ontology consists of two description languages: the first to describe the environment superficially (L1), and the other to give a representation of queries which require inference (L2). In practice the ontology consists of the definitions of tables in a SQL relational database and set queries. I used Microsoft SQL Server 2000 as it allowed me to construct a database with fast and extensive search facilities without having to re-invent the wheel. MS SQL also has many features such as full-text searching, views, and advanced stored procedures. The views allow me to express L1 in an elegant way such that individual queries or stored procedures in L2 depend on, rather than explicitly calling the individual tables. Before outlining the individual languages, I will explain how the ontology represents the user's environment.

Environment events when recorded are inserted into the relational database. This database shapes how we model the environment. The following figure gives the names of the individual tables as well as a brief description on what is stored in the tables:

Table Name	Brief Description
adHocGroup	Relations between ad hoc groups and other objects, description of the group
application	Applications that exist on the computer
applicationCategories	Gives classes to applications such as locally vs remote oriented
category	Container for different categories, intended for documents
conversation	Container for personal communication, i.e. instant messanging

conversationMessage	Container for the individual messages tied to a conversation
email	Container for individual emails
event	Container for all events with references
document	Container for traditional documents
domain	Objects may be explicitly categorized into domains such as neuroscience
file	Container for digital files and folders
link	Objects which are related to each other are 'linked' in different ways
location	2D, 3D and file system locations
keyword	Container for keywords on arbitrary objects
media	Container for metadata on any type of media such as pictures, video, and sound
person	Container for known individuals or groups
query	Stores previous queries from any application
querySpecification	Provides arbitrary fields to link to queries in case the query is more than one string
relationship	Relationships between people, i.e. friend or foe
URL	Container for full URLs and their associated metadata
version	Relates objects to others
webdomain	Container for web domains and their uses

Figure 2: Ontology super-structures

While much of the ontology is general enough to represent a majority of the tasks that occur, certain environments require tailored representations. For example, an informal interview demonstrated that a musician who uses professional audio programs will typically have usage patterns which deviate from the ubiquitous Microsoft Office and web browser user. Audio clips are loaded into Digital Audio Workstation (DAW) projects that may be heavily shared between different projects (such as drum samples), but deducing any nature of these samples by looking at usage patterns will only result in false positives. Many other sets of data which appear to the user as separate entities may all be contained in one file, such as settings for software based synthesizers or MIDI tracks. Our current sets of metadata would not capture the distinction between these embedded file types (as they do not exist separately) and utilize their purposes in

reasoning. Furthermore, many temporary files are created that have varying degrees of significance (which is nearly impossible to guess), that also quite frequently are burned to CD for review outside of any environment which we would monitor.

Part of the power of the ontology is its highly distributed nature, reflecting all the possible relationships that naturally occur. Unfortunately, this also leads to extremely complicated graphs that confused more than explain. For clarity purposes, I will now give examples to explain the design motivations by demonstrating how the ontology is used created in recording of data.

Scenario: Daily e-mail routine

Robert logs into his office computer at the start of his day. His first task is to check his e-mail. As a Linux driver engineer, he subscribes to the Linux Kernel Mailing List. This mailing list is very high traffic and includes many attached files each day. Since this mailing list is general purpose, most of the messages are of little interest to Robert. He is working on an open-source driver, for which he receives emails privately concerning bugs in his drivers as well as links patches. Additionally, he also receives emails from message postings on the company's intranet as well as private emails from his peers, friends, and family.

Robert is an advanced user, and correspondingly sets up his email client to match his profile using explicit information. His email client uses filters to move messages into designated folders. His LKML emails are separated into their appropriate folder by looking at the To address. Any emails originating from the intranet can be moved by

looking at the From address. Since his friends, peers, and family are in his address book tagged by their respective group, their emails are moved into folders according to the From field. Since his company has installed appropriate spam blocking software, Robert assumes any non-filtered emails are from people who are submitting bug reports and patches.

How does the event interpreter appropriately record and annotate his email activity? The following figure shows the stream of events for receiving one email from an unknown person.

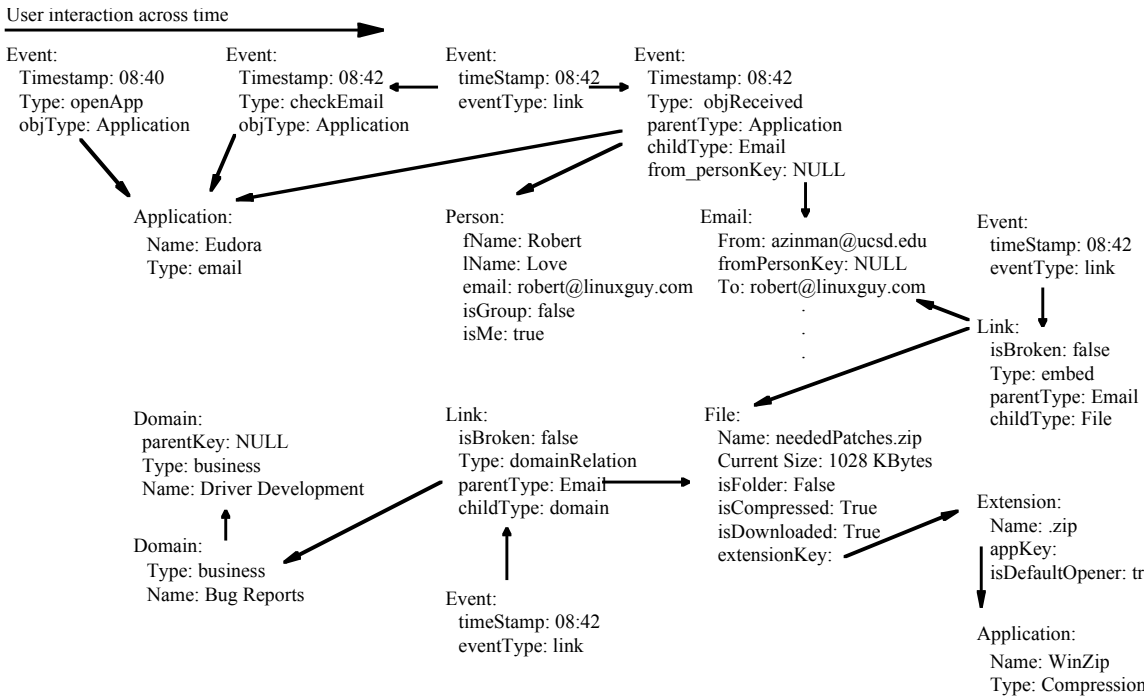

Figure 3: Graph of data stored during when receiving a single of email

Robert opens his email application, and chooses *Retrieve Email*. The interpreter records the opening of the application and uses its own inference rules on the operating system events to pull out the high-level event *checkEmail*. Email is then downloaded to the server, triggering as many *objReceived* as there are emails. *objReceived* was chosen over a more specific term such as *emailReceived* to generalize the concept of receiving an object from another person. Since this person is not in our database, the link to what would be the *from_personKey* is null. However there is a link to Robert who is to receive the object. This decision was guided one of the principals of this ontology: generalize first.

To demonstrate two different possible approaches, let's examine a possible query: "What have I received from Aaron?" Should the ontology have recorded each type of object received in its own event, we would need to know all the types of objects and their corresponding events to answer this question. This method does not scale well, as many queries would need to be modified to accommodate future expansions of the ontology. Instead, we set the type to *objReceived* and specify the following: who it's from, who it's to, what object received it, and where the object is in our system. Should Robert only wish to look at emails received from Aaron, he would restrict his query from the general case to just objects of type email. This also gives us the additional advantage of better understanding the habits of Robert. Since all queries are recorded with their full context, we can see what kinds of information Robert filters out or keeps. Using iterative querying, the general case would display first with other potentially relevant material. While Robert may have had in mind a specific email, the other items returned could provide useful entry points. An *entry point* is "a structure or cue that represents an

invitation to do something – to enter into a new venue or information space" (Kirsh, 2001). Examples of entry points are memos, annotations, new items in an inbox, telephones, and piles of documents.

The actions of either following the link of these entry points or filtering out categories gives us rich information to later analyze. Thus the larger amount of information presented allows Robert to potentially find a miss-categorized item as well as provide richer activity for analysis.

The events of *objReceived* are linked to the event of *checkEmail* so that we can trace back the steps of when it was that Robert actually downloaded those specific lot of email messages. If Robert has large gaps in between mail collection, he may mentally represent the timestamp of each email to when he actually checked it rather than the typical metadata of when the email was sent. Since the email contains an attachment, a file item is created to represent it. Normally file creation events would be tagged with an event for the file creation, but since the file is embedded in the email, Robert is not aware of the file's existence. Queries looking into files would take this into account, allowing Robert to search the files that he created separate from files that are only attached to email. Should Robert actually create the file, the system would create a separate entry for that file. This is necessary because the attachment is embedded into the email, which a saved version is not. However, a link between the saved version and the email version would be created allowing Robert to trace the history of the saved version. Furthermore, once the archive is uncompressed, its contents would be traceable back to the original email through a link to the archive. To make sure this relationship is preserved, the entry

for the saved archive would never be deleted even if Robert deleted it off his computer. This is natural as archives usually serve a temporary function.

Since Robert is a power-user, and was kind enough to explicitly set his own rules of operation for the interpreter, we can classify the domain of the received email. The interpreter would see that azinman@ucsd.edu is an unknown person who is filtered out by any of the email client's rules, leading to the conclusion that this email is bug report. Robert had previously taken the time to create domains and links allowing the ad hoc category "Bug Report" to be linked to this email. Future queries can look in this domain for such files.

The basic sequences of events for emails received are typically similar. In the case of the LKML mailing list, emails received would be attached to an entry under *person* that represents the LKML list (the *isGroup* attribute would be set to *true*). The reason we do not mark the To as Robert is the email was not specifically sent to Robert. This is a much more accurate representation of the email than either disregarding the true author and setting it to LKML or creating two entries for the same email. Furthermore, Robert can query emails according to the LKML membership, individual authors, or authors under the LKML in an intuitive manner.

Not all usages of email are straightforward. Robert's company is using a web-based intranet developed at the Interactive Cognition Lab. Discussions can occur on almost any object in the intranet via email, web, or both. The ontology adapted to fit this situation with little modification allowing Robert to track an entire thread through web forms and email. Discussion using standard email would be handled similarly as noted above assuming the relay does not strip the automatically generated headers which

reference other messages. Message ID headers, however, do not account for web based posts which are forwarded to the appropriate recipients via email. The event interpreter would need to adapt the proprietary references embedded in the emails from the relay which handles web and standard email dialog. Once this is the basis for thread creation, we can attribute Robert's web posting to the current thread. Robert clicks on the provided link in the email to go to the web based form. This URL contains the unique tracking identifier to the current thread. When Robert posts his data, it is first saved in the entry for the URL. It is then possible to create a link of type thread between the URL entry and the entry he chose to reply to.

Having to create specialized event interpreting is a major limitation of the component. All proprietary and nonstandard practices would need to be explicitly coded in order capture the correct activity. Fortunately, one of the strengths of modularizing the ontology is that it does not need to be heavily modified to reflect such changes.

Reasoning

Let's take a closer look at our scenario. Most modern email clients have in the headers a unique message id that allows email threads to be accurately compiled. My ontology supports this by creating a link between email in a thread using the appropriate parent and child relationship. Unfortunately this can create a non-trivial problem: how far do we traverse relationship trees in our searches? Suppose that an email outlining a problem originates in the LKML. Many emails are exchanged about this topic, generating a large thread. This problem leads to other topics, and because most people

tend to hit *Reply* for simplicity, a slightly unrelated topic continues on the original thread. Someone finally comes up with an idea for a solution to the tangent which Robert forwards to his co-worker. His co-worker finally replies with a patch to a different problem which the tangent prompted. Not being immediately relevant, Robert gladly downloads the file for later use. Two weeks later, Robert sees the file but cannot remember what the ill-named file is. He starts up his GUI manager and queries the database to find the source of the file. If we always relied on message-ids then that file would have a link to the original LKML thread, even though semantically it shouldn't.

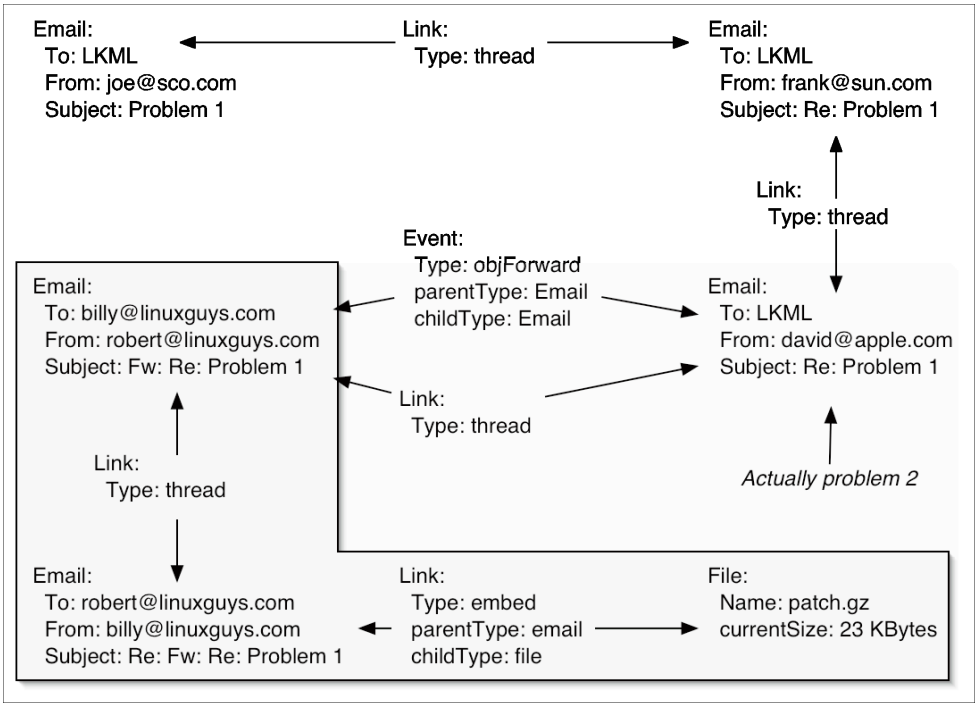

Figure 4: Outlining the source problem

To make matters worse, Robert could have stored mentally and possibly want any part of the chain. How are we to know what return when Robert looks for the *source* of the

document? This is the major problem of scope. We could ignore the problem and simply return everything that is linked to the file. After all, who are we to decide which results are relevant? However if the thread is exceptionally large, this could return too many results. Until we can read minds, it is necessary to have strict definitions in L2. Strict definitions also necessary to create the algorithm that answer the question. With more usage data in future work we can adjust the definitions to provide answers informed by user-center design. For now I have defined *source* using the recursive algorithm for files only listed in Appendix A. The algorithm would stop at the forward. This distinction was chosen because when something is forwarded it goes outside of the mailing list, which in many cases changes the semantics.

There are limitations, however. Besides the need to update the definition for new programs and their relations, certain events would not get properly captured. For example, in the MS SQL Enterprise Manager I can copy the contents of a remote database onto my machine. The file that would be created contains the entire database without separating its actual content into their appropriate function types. Since this file is behind the scenes, its creation bares no significance to the user. However if we were to monitor without any intelligence when an application creates a file, we would note the file creation which the user is completely ignorant of. Furthermore, when save a local version of a stored procedure from the copied database, there is no linkage to the original remote script. In order to create the meaningful link, we would need to have our event interpreter understand the different types of actions within Enterprise Manager to a high degree. Therefore the system needs to have its event interpretations hand-crafted for each type of application to make sure significant events are processed appropriately.

Furthermore, this example demonstrates that competently automated ontological learning could not occur as the computer has no way of understanding which events are significant. With understanding context such a high priority for correct annotation of events, it is necessary to have an expert system base over a purely *Tabula Rosa* connectionist system.

Even though the reasoning engine will never be fully complete, it remains an essential piece of the puzzle that needs substantially more work to empower the visualization tools. However, the original goals of implementing a large L1 and limited L2 have been met. The current L2 has focused on providing useful strict definitions for baseline queries. Since we cannot predict what queries will actually get used, future work is targeted at focusing on fuzzy inferences of categorizing activity to allow for things such as predicting what is related to project X.

Visualization

Initial work towards a GUI manager was performed using Java and Multivalent. Multivalent is a Java framework which creates a browser capable of rendering standard text, HTML pages, and PDFs. Additionally Multivalent supports direct annotation on documents it renders: an event and data the system could record and incorporate.

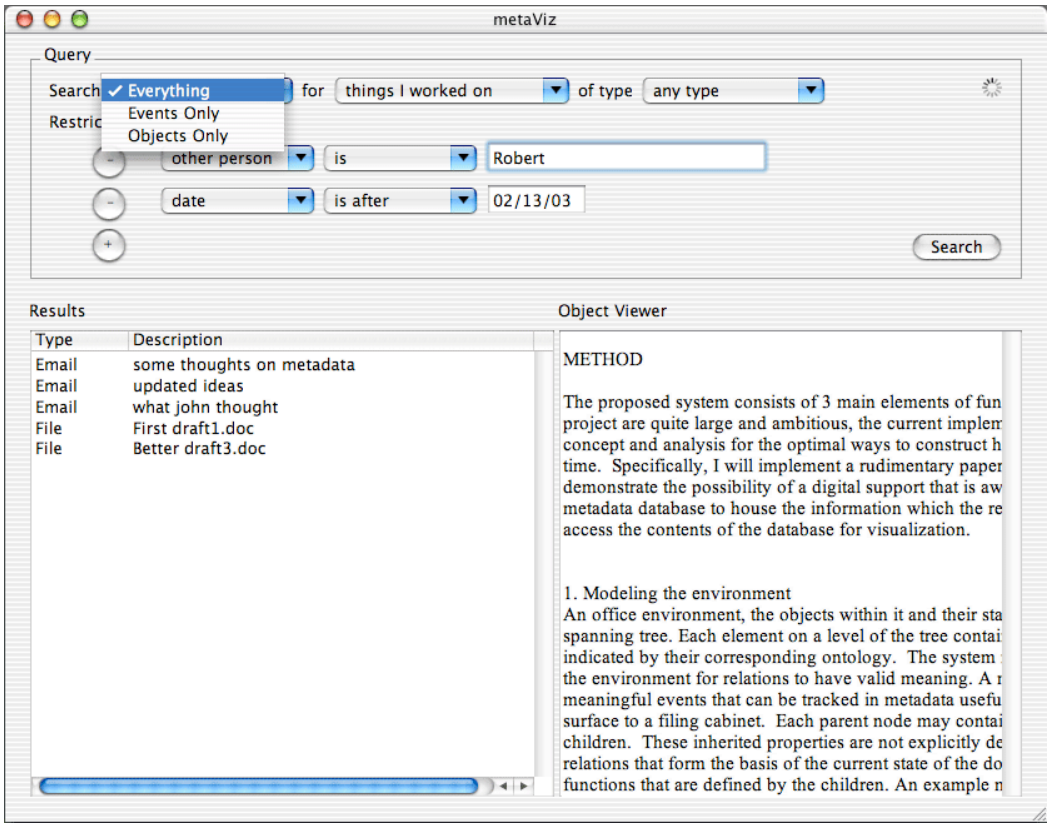

Figure 5: Prototype of metaViz

metaViz, the GUI manager, is very limited in its usable functionality. Unfortunately frameworks for GUI programming such as Swing are very time consuming to build interfaces, let alone incorporate function. Most of the system was tested by creating SQL queries by hand to test the stored procedures and views. metaViz remains a major project to work on for the future, but its specification has already been established.

metaViz has to find a balance between ease of use and power. By creating direct SQL queries of arbitrary size, almost any type of relationship can be assessed. This leads to the question: how can we let the user query any type of relationship they wish to assess with minimal cost? This has been attempted to be solved using several key features: iterative querying, play lists, and arbitrary restrictions on L1 and L2.

Iterative querying refers to instantly displaying the query as it is entered. As soon as the user has selected a major category of what they are trying to find, the results section would fill up with all possible answers. Typically this will be a huge list, but as the user proceeds it allows for viewing of how the search space is being restricted. Furthermore, it allows for entry points to potentially be explored that might otherwise be left out. Apple has a simple but effective iterative querying mechanism on its iTunes jukebox software. A search box shown above the entire music library allows for the user to search all metadata for a particular string. With each letter, the library reduces its contents to reflect the query.

Play lists refers to a HCI concept which Apple in particular is using heavily in its software lately. metaViz uses this concept as a way to store arbitrary data in an ad hoc structure which the user can access at any time. The play list may contain an actual query which would be rendered live with every access. Play lists could also contain collections of items such as objects and events. This leads to a successful method of capturing personal metadata in pure way which reflects the ontology, which is very significant for the goals of the system. Additionally, play lists can be temporary or permanent between sessions allowing users to have a workspace to explore their environment. Play lists can also be used in searches where they become the master set which is searched, one of the type of restrictions that may be placed.

In order to express a query, the user needs to be able to put restrictions on the results that are returned. Possible restrictions could include directly accessing attributes of the ontology such as “file created after 2/03/02” using L1 or access the results of L2. Due to the nature of L2, it can be difficult to have the restrictions the user intends reflect

what is actually restricted. For example, when attempting to find the source of a document, many different items could be returned as in our scenario. A possible solution is to not allow for L2 queries which return more than one entry. In order to answer the query the user would need to use successive queries through play lists. Unfortunately this may not be natural, resulting in the need for user surveys to inform better design.

Lastly metaViz needs to support spatial querying. The system tracks the 2D locations of files as they move across windowing operating environments. Spatial cues are very powerful in helping a user answer a query, especially because people tend to cluster their desktops in ad hoc categories according to various functions.

DISCUSSION

While its nice that Robert has a simple, well defined environment, most users do not. How can we cope with the chaotic nature of email, attaching the correct corresponding domain to objects in an ad hoc fashion? Is it possible to analyze activity in a way that auto-magically produces correct ad hoc groups and categories?

It is clear that each user has different demands and uses of their environment. I have attempted to create an ontology which generalizes any user's environment for much of their core activity. However, several major problems that will always plague the system.

There is the problem of scope: it is difficult to know where to stop following links to get our answer. While to the computer it all may seem relevant, the user really wants a particular point in search space. This can be tackled by setting arbitrary thresholds as

well as strict definitions of queries. A much better way is creating visualization software which minimizes the cost of wadding through large sets of results as well as hijacking a real brain in navigating through search space.

We have the problem of semantics in natural language queries, where we need strict definitions that can handle many different contexts of problems. This can be approximated by using a small vocabulary (L1 and L2) to search instead of pure natural language queries. Yet using a restricted vocabulary only partially addresses the issue of context. If we could accurately segment types of usage such as the project being worked on or web browsing, we would be able to provide better results by trimming what we know the user isn't looking for. In order to understand context we need better theory on the underlying cognitive functions of personal metadata as well as office ecologies.

All is not lost. The World Wide Web and its demand for better search tools has prompted much research into the area of text categorization. Work by Wang among many others have shown that algorithms can exist which meaningfully classify sets of data. My ontology supports automated classification through adHocGroups marked machine generated. Their categories could appear as restrictions or play lists which to base queries off. The system can significantly benefit when metaViz can provide an interface to correct misclassifications, link classifiers to domains, or manual creation of categories. Most importantly, the current ontology and descriptive language provides a large interface to search metadata that would not be possible any other way. The system captures and annotates many types of activity, providing the ability to trace many objects throughout their life history imposed by the user's projected personal metadata.

FUTURE WORK

This project has outlined a clear strategy for understanding and supporting office activity. However, it is dependent on countless more hours of follow up research for the envisioned system to be truly realized. The main component needed to propel future research is capturing and analyzing natural strategies in the office ecology. We need a clear answer on what kinds of ad hoc categories exist in the digital domain, and how they extend or differ from their physical counterparts. It is necessary to know how people cope with overload of files, and what strategies are used for spring cleaning. Are there varying degrees of organization? How do people decide where to place intermittent files, and can we predict temporal relationships using available data?

Even without a more complete theory, more of the system can and will be implemented. The first task is to create an application which can effectively monitor and interpret digital activity instead of hand-coding transcripts. Working on an event interpreter will impact the ontology, changing to accommodate unforeseen relationships and expanding to better represent the environment. Lastly, metaViz needs to be fully implemented at least rudimentarily, in addition to exploration of better methods of visualization.

Current work involves extending the reasoning engine in an attempt to cluster files according to activity. Temporal relationships are extremely important in determining context as well as window management. For example, constant switching between two applications or positioning of windows such that two objects are visible at the same time is very meaningful personal metadata. It is hypothesized that windows of

time can be analyzed according to activity of files (how often a file is accessed, etc) in conjunction with the activity of applications to provide analogous to physical clustering techniques such as piling or stacking. By assigning activity as a relative value, irrelevant objects may be accurately discounted. It is this type of analysis that provides the user with powerful tools for search as well as the cognitive researcher methods of understanding how people work.

APPENDIX A.

Pseudo-code for finding the *source*

```
FUNCTION findSource( ourFile : = file to search )
      // initialize the data structure we are returning
      initialize(toReturn)

      // check to see if this file belongs to a user-generated
      // ad hoc group.
      IF EXISTS( link WHERE child = ourFile
                              parent = adHocGroup
                        AND
                        WHERE adHocGroup.machineGenerated IS
                               FALSE
                )
      THEN
            toReturn := toReturn + adHocGroup
      END IF

      // check to see if this is an uncompressed file
      IF EXISTS( link WHERE child = ourFile
                              parentType = file
                              type = embed
                        AND
                        WHERE parent.isCompressed IS TRUE
                )
      THEN
            toReturn := toReturn + link.parent
      END IF

      // check to see if this file is embedded in anything
      // we don't want to keep on searching the source of
      // what its embedded in because we don't know how
      // significant that may be, unless its an email
      IF EXISTS( link WHERE child = ourFile
                              type = embed
                        AND
                        WHERE IF ( parentType = file THEN
                                      parent.isCompressed IS FALSE
                                 )
                )
      THEN
            toReturn := toReturn + link.parent
            IF ( parent.Type = email )
            THEN
                  toReturn := toReturn+scourEmails(link.parent, 0)
```

```
      END IF
END IF


// see if this file is another version of an existing
// file, return it and its source
otherVersion := TRUE
IF EXISTS ( link WHERE child = ourFile
                       parentType = file
                       type = version
          )
THEN
      toReturn := toReturn + link.parent
      toReturn := toReturn + findSource(link.parent)
      otherVersion := TRUE
END IF


// look at what application created to understand what we
// should be looking for

appCreator := ourFile.appCreator
IF EXISTS ( appCategory WHERE
                        appCategory.appType=appCreator.type
          )
THEN
      category := appCategory
END IF

IF ( category IS NOT NULL AND category = "Create Locally" )
THEN
      // we know our file comes from a local source, so
      // we don't need to try and follow links to get to
      // a remote object.

      // if there is a parent version then search only that
      IF ( otherVersion IS FALSE )
            IF EXISTS ( event WHERE type = objCreated
                                    parentType = application
                                    child = ourFile
                      )
            THEN
                  toReturn := toReturn + event
            END IF
      END IF
ELSE
      // either it comes from a remote source or we don't
      // know the source... either way we need application
      // specific searches.

      // check to see if it came from an instant message
      IF ( appCreator.type = "Instant Message" )
```

```
THEN
      // find the conversation that it originated from
      IF EXISTS ( link WHERE child = ourFile
                              parentType = conversation
                )
      THEN
            toReturn := toReturn + link.parent
      END IF
END IF

// we do not need to check to see if it was from
// and email here.  this is because we already have
// looked for other versions recursively, and if they
// exist, then it will eventually find the embedded
// version in the source email, which is handled above

// if we got this from the web, it could potentially
// result from a link in an email or standalone
IF ( appCreator.type = "Web Browser" )
THEN
      // we need to return the web page it originated
      // from as well as any query which led to it.
      IF EXISTS ( link WHERE child = ourFile
                              parentType = URL
                              type = downloadedFrom
                        AND
                        WHERE URL.isWebPage IS TRUE
                )
      THEN
            webpage := link.parent
            toReturn := toReturn + webpage
            toReturn := toReturn+findQuery(webpage)

            // check to see if this web page was result
            // of an email link
            IF EXISTS ( link WHERE child = webpage
                                    parentType = email
                        )
            THEN
                  toReturn := scourEmail(link.parent, 0)
            END IF
      END IF
END IF

// if we got it from an ftp session, return the URL
// to the remote file
IF ( appCreator.type = "FTP Client" )
THEN
      IF EXISTS ( link WHERE child = ourFile
                              parentType = URL
                              type = downloadedFrom
                        AND
```

```
                                    WHERE URL.isFTP IS TRUE
                              )
                  THEN
                        toReturn := toReturn + link.parent
                  END IF
            END IF
      END IF

      RETURN toReturn
END FUNCTION

FUNCTION scourEmails( ourEmail := email to search, i:=iteration)
      // we only return up to 20 links up in the thread to
      // prevent too much irrelevant information.  since its
      // a variable, we can later adjust this
      IF ( i = MAX_EMAIL_ITERATIONS )
      THEN
            RETURN NULL
      END IF

      initialize(toReturn)
      toReturn := ourEmail

      // check to see if this current email was forwarded..if so
      // stop there
      IF EXISTS ( event WHERE type = objForward
                              child = ourEmail
                              parentType = email
                  )
      THEN
            toReturn := toReturn + event.parent
            RETURN toReturn
      END IF

      // otherwise check to see if this is a member of a thread
      IF EXISTS ( link WHERE child = ourEmail
                             parentType = email
                             type = thread
                  )
      THEN
            toReturn := toReturn + scourEmail(link.parent, i + 1)
      END IF

      RETURN toReturn
END FUNCTION

FUNCTION findQuery( ourURL := web page to search )
      // first see if a query has been tagged to this web page
      // if so, return it.  otherwise keep on going up in the
      // history to find any other query should it exist.

      IF EXISTS ( link WHERE child = ourURL
```

```
                              type = resultOfQuery
                              parentType = query
                    )
      THEN
            RETURN link.parent
      ELSE
            IF EXISTS ( link WHERE child = ourURL
                                    parentType = URL
                                    type = href
                        )
            THEN
                  RETURN findQuery(link.parent)
            ELSE
                  RETURN NULL
            END IF
      END IF
END FUNCTION
```

APPENDIX B.

List of questions that inspired part of the ontological design:

Where are my documents that I worked on with Robert

Where are the files Robert gave me?

What did I work on in this month?

What were the search queries that produced the PDFs that ended up in the pervious query?

What emails contained those keywords?

What document contains these words … in this Boolean query … ?

Who were the authors that wrote the papers that contains these words …

What are all the web pages that I pulled these images from?

Where are the files that have been written from an encryption type application?

Who has manipulated this file … ?

What other documents has this person … manipulated in this time frame … ?

Of those in the previous query, which are in this project … as defined by this play list … ?

Where's my outline on this subject … ?

What functions deal with the aspect for text output in this source code? (NOT POSSIBLE)

Where's this file from?

What has been in this location … on my desk with this time frame … ?

Who has modified or accessed this document … ?

What is marked to do?

What email has resulted in the creation of a word document? (NOT POSSIBLE since we cannot predict intentions without a direct link)

What email has come from John?

What is related to this file … ?

What corresponds to the following keywords … ?

Where did this document come from? (Question is attempting to figure out if it originated in the physical or digital domain)

Where are all the copies or versions of this document … ?

What are all the files I accessed in this time span … ?

What attachments haven't I opened?

Where are all my notes on this book?

What files resulted from the google search using this query … ?

Who's documents are in this directory … ?

REFERENCES

Al-Kofahi, K. Tyrrell, A. Vachher, A. Travers, T. Jackson, P. (2001) Combining multiple classifiers for text categorization. Proceedings from the International Conference on Information and Knowledge Management, Proceedings 2001. pp 97-104.

Avello, D. Gutierrez, D. (2002) The cooperative web: A complement to the semantic web. Proceedings from IEEE Computer Society's International Computer Software and Applications Conference 2002. pp 179-183.

Barsalou, L. Solomon, K.O. (2001). Representing properties locally. Cognitive Psychology, 43, pp. 129-169

Barsalou, L. (1983). Ad hoc categories. Memory & Cognition, vol 11 (3), pp. 211-227.

Bruggemann, B., Holz, K., Molkenthin, F. (2000) Semantic documentation in engineering - Content retrieval by arbitrary information. Computing in Civil and Building Engineering. v 2 2000. pp 828-835.

Fensel, D. Van Harmelen, F. Horrocks, I. McGuinness, D L. Patel-Schneider, P F. (2001) OIL: An ontology infrastructure for the semantic web. IEEE Intelligent Systems & Their Applications. v 16 n 2 March/April 2001. pp 38-45

Grumbach, S. Tininini, L. (2000) Automatic aggregation using explicit metadata. Scientific and Statistical Database Management - Proceedings of the International Working Conference 2000. IEEE, Los Alamitos, CA, USA.. pp 85-94.

Jackendoff, R. (1987) *Consciousness & the Computational Mind*. MIT Press, pp. 193-212.

Kirsh, D. (2001). The Context of Work, Human Computer Interaction, (forthcoming)

Kirsh, D. (1995). The Intelligent Use of Space. Artificial Intelligence, Vol. 73, Number 1-2, pp. 31-68

Maedche, Alexander D. (2002) Ontology learning for the semantic Web. Kluwer Academic Publishers, Boston.

Marchand, E. Chaumette, F. (2002) Virtual visual servoing: A framework for real-time augmented reality. Computer Graphics Forum. v 21 n 3 September 2002. pp 289-297.

McCloskey, M. Glucksberg, S. (1978). Natural categories: Well defined or fuzzy sets? Memory & Cognition, vol 6. (4), pp. 462-472.

Mennis, J.L. (2000) Human cognition as a foundation for GIS database representation. In: Graduate Student Research Papers, UCGIS Summer Assembly, June 21-24, 2000, Welches, OR. Leesburg, VA: University Consortium for Geographic Information Science, pp. 4.1-4.17.

Mennis, J.L., Peuquet, D.J., and Qian, L. (2000) A conceptual framework for incorporating cognitive principles into geographical database representation. International Journal of Geographical Information Science, 14(6). pp. 501-520.

Rais-Ghasem, M. Corriveau, J.P. (1996). Beyond Concept Recognition. Ottawa, Ont., Canada.

Raskar, R., Welch, G., Cutts, M., Lake, A., Stesin, L., Fuchs, H.(1998). The Office of the Future : A Unified Approach to Image-Based Modeling and Spatially Immersive Displays. ACM SIGGRAPH, Orlando FL.

Raskar, R., Welch, G., Cutts, Stuerzlinger, W. (1998). Efficient image generation for multiprojector and multisurface display surfaces". Ninth EuroGraphics Rendering Workshop, June 1998 (Appeared in Drettakis, G., Max, N. (eds.), Rendering Techniques '98. Proceedings of the Eurographics Workshop in Vienna, Austria.)

Savage-Rumbaugh, E.S et al. (1986) Spontaneous symbol acquisition and communicative use by pygmy chimpanzees (Pan paniscus). Journal of Experimental Psychology: General. vol 115. pp. 211-235.

Wang, Y., Tan, C., Lee, C. (2002) The Use of Bigrams to Enhance Text Categorization.

Whittaker, S. Hirschberg, J. (2001). The character, value and management of personal paper archives. AMC Transactions of Computer Human Interaction, vol 8, pp. 150-170.

Wilensky, Robert (2001). The Multivalent Browser: A Platform for New Ideas. Proceedings of Document Engineering 2001. Atlanta, Georgia.