SemanticOrganizer: A Customizable Semantic Repository for Distributed NASA Project Teams

Richard M. Keller[1], Daniel C. Berrios[2], Robert E. Carvalho[1], David R. Hall[3], Stephen J. Rich[4], Ian B. Sturken[3], Keith J. Swanson[1], and Shawn R. Wolfe[1]

[1]Computational Sciences Division, NASA Ames Research Center, Moffett Field, CA
{rkeller, rcarvalho, kswanson, swolfe}@arc.nasa.gov
[2]University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA
dberrios@arc.nasa.gov
[3]QSS Group, Inc. NASA Ames Research Center, Moffett Field, CA
{dhall, isturken}@arc.nasa.gov
[4]SAIC, NASA Ames Research Center, Moffett Field, CA
srich@arc.nasa.gov

Abstract. SemanticOrganizer is a collaborative knowledge management system designed to support distributed NASA projects, including diverse teams of scientists, engineers, and accident investigators. The system provides a customizable, semantically structured information repository that stores work products relevant to multiple projects of differing types. SemanticOrganizer is one of the earliest and largest semantic web applications deployed at NASA to date, and has been used in diverse contexts ranging from the investigation of Space Shuttle Columbia's accident to the search for life on other planets. Although the underlying repository employs a single unified ontology, access control and ontology customization mechanisms make the repository contents appear different for each project team. This paper describes SemanticOrganizer, its customization facilities, and a sampling of its applications. The paper also summarizes some key lessons learned from building and fielding a successful semantic web application across a wide-ranging set of domains with diverse users.

1 Introduction

Over the past five years, the semantic web community has been busily designing languages, developing theories, and defining standards in the spirit of the vision set forth by Berners-Lee [1]. There is no lack of publications documenting progress in this new area of research. However, practical semantic web applications in routine daily use are still uncommon. We have developed and deployed a semantic web application at NASA with over 500 users accessing a web of 45,000 information nodes connected by over 150,000 links. The SemanticOrganizer system [2] has been used in diverse contexts within NASA ranging from support for the Shuttle Columbia accident investigation to the search for life on other planets; from the execution of Mars mission simulations to the analysis of U.S. aviation safety and study of malarial disease in Kenya. This paper describes our system and some of the practical challenges of build-

ing and fielding a successful semantic web application across a wide-ranging set of domains. One of the key lessons we learned in building a successful application for NASA is to understand the limits of shared ontologies and the importance of tuning terminology and semantics for specific groups of users performing specific tasks. We describe our methods, compromises, and workarounds developed to enable maximal sharing of our ontology structure across diverse teams of users.

SemanticOrganizer is a collaborative knowledge management application designed to support distributed project teams of NASA scientists and engineers. Knowledge management systems can play an important role by enabling project teams to communicate and share information more effectively. Toward this goal, SemanticOrganizer provides a semantically-structured information repository that serves as a common access point for all work products related to an ongoing project. With a web interface, users can upload repository documents, data, images, and other relevant information stored in any of a wide variety of file formats (image, video, audio, document, spreadsheet, project management, binary, etc.). The repository stores not only files, but also concepts that have no associated electronic manifestation, such as hypotheses, field sites, and engineered systems. Hardware or software systems that generate data used by a project team can access the repository via an XML-based API.

Although there are many document management tools on the market to support basic information-sharing needs, NASA science and engineering teams have some specialized requirements that justify more specialized solutions. Examples of such teams include scientific research teams, accident investigation teams, space exploration teams, engineering design teams, and safety investigation teams, among others. Some of their distinctive requirements include:

- *sharing of heterogeneous technical information*: teams need to exchange many types of specialized scientific and technical information in various formats;
- *detailed descriptive metadata*: teams use a precise technical terminology to record detailed characteristics relating to information provenance, quality, and collection methodology;
- *multi-dimensional correlation and dependency tracking*: teams need to interrelate and explore technical information along a variety of axes simultaneously and rapidly make connections to new information;
- *evidential reasoning*: teams must be able to store hypotheses along with supporting and refuting facts, and methodically analyze causal relationships;
- *experimentation*: teams must carry out experiments to test hypotheses with systematic measurements;
- *instrument-based data production*: teams use specialized scientific instruments and sensors as data sources;
- *security and access control*: information being collected and analyzed may be highly proprietary, competitively sensitive, and/or legally restricted; and
- *historical record:* project teams must document their work process and products – including both successes and failures – for subsequent scrutiny (e.g., to allow follow-on teams to validate, replicate, or extend the work, to capture lessons learned, or to satisfy legal requirements).

Aside from the above requirements, we faced several other major technical challenges in building the SemanticOrganizer repository system. One of the most difficult challenges was to make the information easily and intuitively accessible to users, even

when different teams employ different terms, relationships, and models to mentally organize their work products. Rather than organizing information using generic indexing schemes and organizational models, we felt it was important to employ terms, concepts, and natural distinctions that make sense in users' own work contexts. A second and related challenge was to develop a single application that could be rapidly customized to meet the needs of many different types of teams simultaneously. Many of the candidate user teams were as small as just two or three people, so they could not afford the overhead of running their own server installation or handling system administration. Thus, the system had to be centrally deployed while still being customized for each teams' distinctive work context. A third key challenge involved knowledge acquisition and the automatic ingestion of information. With the large volume of information generated during certain projects and the complexity of the semantic interrelationships among information, users cannot be expected to maintain the repository without some machine assistance. A final challenge is providing rapid, precise access to repository information despite the large volume of information.

We found that a semantic web framework provided a sound basis for our system. Storing information in a networked node and link structure, rather than a conventional hierarchical structure, addressed the need to hyperconnect information along multiple dimensions. Using formal ontologies provided a customizable vocabulary and a structured mechanism for defining heterogeneous types of information along with their associated metadata and permissible relationships to other information. We employed an automated inference system to automatically maintain knowledge in the repository. However, we also found it necessary to add a host of practical capabilities on top of the basic semantic web framework: access control mechanisms for network-structured information, authentication and security, ontology renaming and aliasing schemes, effective interfaces for accessing semantically-structured information, and APIs to enable ingestion of agent-delivered information.

The balance of the paper is organized as follows. Section 2 describes the basic SemanticOrganizer system in more detail. Section 3 describes the mechanisms we have developed to customize the system for multiple groups simultaneously. Section 4 discusses related work. Section 5 highlights NASA applications that we have developed using SemanticOrganizer and explains extra functionality that was added to support specific application needs. Section 6 summarizes some of the lessons learned from our experience building a practical semantic web application and Section 7 concludes.

2 The SemanticOrganizer System

SemanticOrganizer consists of a network-structured semantic hypermedia repository [3] of typed *information items*. Each repository item represents something relevant to a project team (e.g., a specific person, place, hypothesis, document, physical sample, subsystem, meeting, event, etc.). An item includes a set of descriptive metadata properties and optionally, an attached file containing an image, dataset, document, or other relevant electronic product. The items are extensively cross-linked via semantically labeled relations to permit easy access to interrelated pieces of information. For example, Figure 1 illustrates a small portion of a semantic repository that

was developed for a NASA accident investigation team. The item in the center of the diagram represents a rotor assembly system being testing in a wind tunnel. The links between items indicate that the rotor assembly was operated by John Smith, who is being investigated by the CRW (Canard Rotor Wing) investigation. Rotor fatigue was observed and is a hypothesized causes of the mishap. Fatigue is documented by evidence consisting of a metallurgy report and a scanning electron microscope (SEM) image. These types of items and relationships are natural and appropriate for this domain, whereas others would be required to support a different type of team.

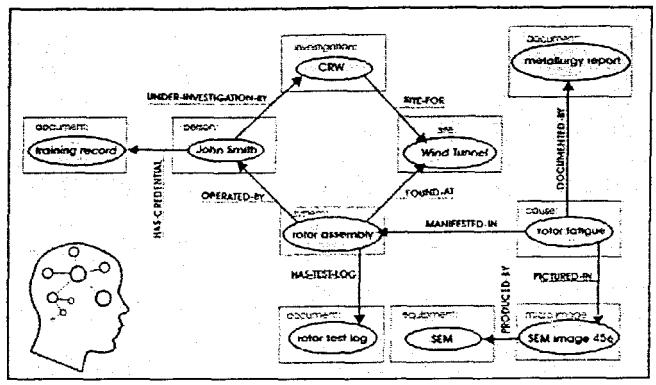

Fig. 1. Portion of semantic repository item network for CRW accident investigation

A master ontology (Figure 2) describes all the different types of items for SemanticOrganizer applications, and defines links that can be used to express relationships between the items. (In this paper, we use the term *item type* and ontology *class* interchangeably; similarly, *item* and *instance* are interchangeable.) A *link* or *relation* is defined by specifying its name and its domain and range classes, along with the name of its reverse link. (All links are bidirectional.) We began development of SemanticOrganizer in 1999, prior to the standardization of semantic web languages; as a result, the system was built using a custom-developed representation language. Our language has the equivalent representational power of RDFS [4], except that it does not permit the subclassing of relationships.

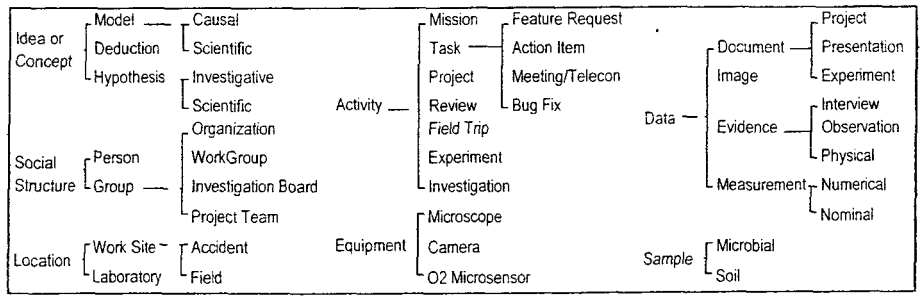

Fig. 2. Representative classes from SemanticOrganizer's master ontology. The entire ontology has over 350 classes and reaches a maximum depth of six.

SemanticOrganizer is built in Java and its ontology classes and instances are stored in a MySQL database. The system includes an inference component that is built on top of Jess [5]. Explicit rules can be defined that operate on the ontology and create or modify items/links in the repository or establish property values.

SemanticOrganizer includes an email distribution and archiving facility that allows teams to create ad-hoc email lists. Email sent to a SemanticOrganizer distribution list is forwarded to recipients and archived as an email message item within the team's repository. Attachments are preserved along with the message body, and instances representing the sender and recipients are automatically linked to the message. A more experimental system component under development is the Semantic Annotator, which parses text documents, such as email messages, and links them to relevant items in the repository. The Semantic Annotator employs WordNet [6], as well as other sources of information, to select relevant items for linking. (The specific algorithm used by the Semantic Annotator is beyond the scope of this paper.)

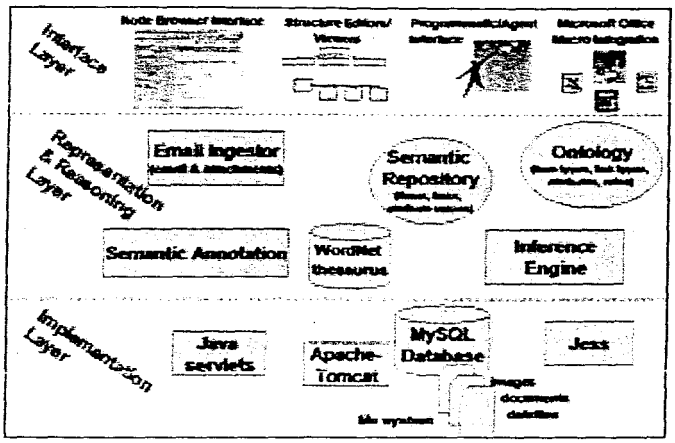

Fig. 3. SemanticOrganizer's architectural components

SemanticOrganizer's various components are depicted in Figure 3. For conceptual clarity, in the diagram we distinguish between the ontology, which stores the class and link types, and the semantic network repository, which stores the interlinked instances. In practice, these components are implemented using a single representational mechanism that stores both classes and instances. Although the repository is stored on a single server, access control and ontology customization mechanisms make the repository format and content appear different for each group of users. In essence, SemanticOrganizer is a set of *virtual repositories*, each built upon the same representational framework and storage mechanisms, yet each custom-designed to suit the needs of its individual users. The customization process is described in Section 3.

SemanticOrganizer users enter and interlink items using a servlet-driven Web interface that enables them to navigate through the semantic network repository, view metadata and files, and search for specific items (see Figure 4). The interface also allows users to create and interlink items, upload files and attach them to items, and

flexibly search through the repository. The core interface uses only HTML and basic JavaScript to maximize compatibility with standard browsers. Aside from the HTML-based Web interface, the system also includes some specialized applets for visualizing and editing specific interlinked structures of items. (A more general graphical network visualization component is currently under development.) SemanticOrganizer features an XML-based API that enables external agents to access the repository and manipulate its contents. In addition, we have developed a set of Visual Basic macros that provide an interface between Microsoft Office documents and SemanticOrganizer using the Office application's menu bar.

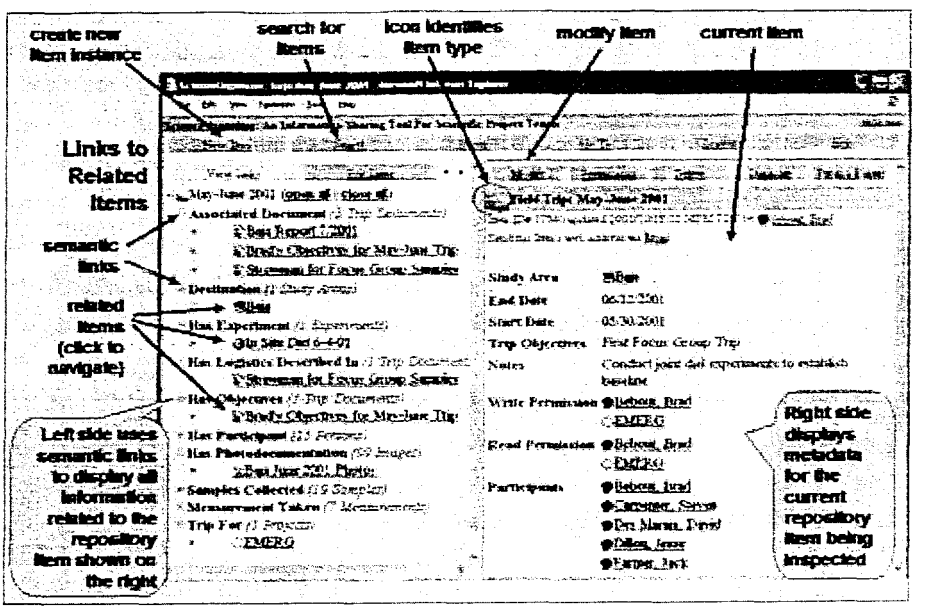

Fig. 4. SemanticOrganizer's Web interface displaying a scientific 'field trip' item at right. Note individual and group read and write permissions for the item. Links to related items are displayed at left.

Security and authentication are handled by HTTPS encryption and individual user logins. No access is permitted to users without an assigned login as part of one or more established project teams. Once inside the repository, user access to items is controlled by a permission management system. This system limits users' access to a defined subnet within the overall information space that contains information relevant to their team. As part of this access control system, each instance in the repository has a set of read and write permissions recording the individual users or sets of users (i.e., groups) that can view and modify the instance.

A set of successively more sophisticated search techniques is available to SemanticOrganizer users. A basic search allows users to locate items by entering a text string and searching for matching items. The user can specify where the match must occur: in an item name, in a property value for an item, or in the text of a document attached to an item. In addition, the user can limit the search to one or more item types. An in-

termediate search option allows the user to specify property value matching requirements involving a conjunction of constraints on numeric fields, enumerated fields, and text fields. Finally, a sophisticated semantic search is available for matching patterns of multiple interlinked items with constraining property values [7].

3 Application Customization Mechanisms

SemanticOrganizer is specifically designed to support multiple deployments across different types of distributed project teams. Knowledge modelers work with each new group of users to understand their unique requirements. The modelers add or reuse ontology classes to form a custom application suitable for the team. To encourage reuse of class, property, and link definitions, the system contains a single unified ontology that addresses the needs of users involved in more than 25 different project teams. Each of these teams uses only a subset of the classes defined in the ontology. Ontology classes are assigned to users through a process illustrated in Figure 5.

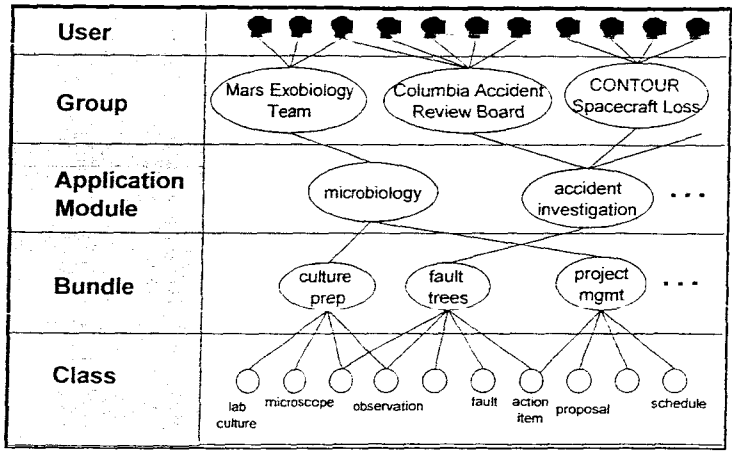

Fig. 5. Mapping ontology classes to users via bundles, application modules and groups

At the lowest levels, classes are grouped into *bundles*, where each bundle defines a set of classes relevant to a specific task function. For example, all of the classes relevant to growing microbial cultures (e.g., physical samples, microscopes, lab cultures, culturing media) might constitute one bundle; all classes relevant to project management (e.g., project plans, project documents, funding sources, proposals, meetings) might be another bundle. Aside from grouping related classes, bundles provide a mechanism for aliasing classes to control their interface presentation to users. For example, the ontology includes a class called 'field site'. A field site is simply a location away from the normal place of business where investigation activities are conducted. Although there may be a general consensus about this definition across different application teams, the terminology used to describe the concept may differ. For example, whereas geologists may be perfectly comfortable with the term 'field site', acci-

dent investigators may prefer the term 'accident site'. Although this distinction may seem trivial, employing appropriate terminology is essential to user acceptance. The bundling mechanism allows domain modelers to alias classes with a new name. (Note that renaming of properties is not supported, at present, but would prove useful.)

At the next level up in Figure 5, sets of bundles are grouped together as *application modules*. These modules contain all the bundles that correspond to relevant tasks for a given application. For example, there might be a microbiology investigation team growing microbial cultures as part of a scientific research project. In this case, the application builder would simply define a module that includes the microbial culture bundle and the project management bundle. At the top levels of Figure 5, modules are assigned to groups of users, and finally through these groups, individual users gain access to the appropriate classes for their application. A user can be assigned more than one module if he or she is involved in more than one group. For example, a microbiologist involved in the Mars Exobiology team may also be on the Columbia Accident Review Board as a scientific consultant. Note that this discussion explicitly covers assignment of ontology classes, not ontology relations, to users. However, the assignment of relations can be considered a byproduct of this process. A specific relation is available to a user if and only if its domain and range classes are available.

4 Related Work

We have identified four categories of Web-based systems that share important characteristics with SemanticOrganizer: conventional Web portals, content/document management systems, semantic portals, and semantic repositories. Conventional Web portals, as exemplified by sites such as MyYahoo, typically allow users to selectively subscribe to various published content and customize its presentation. Commercial content/document management systems (e.g., Documentum, FileNet, Vignette, and DocuShare) are more focused on supporting daily work processes than publishing. They allow users to upload, store, and share content (including intermediary work products). To summarize the difference, portals are intended to publish finished content, whereas document management systems manage transient and unfinished work products that are not necessarily appropriate for external or internal publication. Neither type of system is semantically based.

Semantic portals [8-12] and semantic repositories [13, 14] can be viewed as analogous to "regular" portals and content management systems, except that they use an underlying ontology to enhance their content with semantics. As a generalization, the primary difference between them is that semantic portals are intended to publish finalized information, whereas semantic repositories are intended to manage work products in process. SemanticOrganizer is a prime example of a semantic repository; it is intended to provide semantics-enhanced content management support across various phases of a project lifecycle.

Worthy of special note is ODESeW [8], which has many features in common with SemanticOrganizer. ODESeW is primarily a semantic portal but also allows users to edit both the underlying ontology and its instances. Access can be granted (or denied) at the instance level or the class level, as well as the ability to instantiate in-

stances of a specific class. Unlike SemanticOrganizer, these permissions are controlled solely by an administrator, and only read access can be denied for a particular instance, not write access. Furthermore, although the ability to hide all instances of a class is similar to the SemanticOrganizer's customization mechanism, there is no facility to alter the presentation of the *same* instance to different user groups.

5 Applications

5.1 Background

With over 500 registered users and over a half-million RDF-style triples in its repository, SemanticOrganizer is one of the largest semantic web applications that has been fielded at NASA to date. The system was first deployed in 2001 to support a small group of collaborating research scientists. As of April 2004, over 25 different collaborating groups – ranging in size from 2 people to over 100 – have used SemanticOrganizer in conjunction with their projects. System users are drawn from more than 50 different organizations throughout NASA, industry, and academia. The overall ontology contains over 350 classes and over 1000 relationships. Over 14,000 electronic file attachments have been uploaded into the system and more than 12,000 email messages have been distributed and archived.

SemanticOrganizer has found application primarily within two user communities: the NASA scientific community (where the system is known as *ScienceOrganizer*), and the NASA safety and accident investigation community (where the system is known as *InvestigationOrganizer* or *IO*). In the following sections, we describe prototypical applications within these distinct SemanticOrganizer user communities.

5.2 ScienceOrganizer

ScienceOrganizer was originally developed to address the information management needs of distributed NASA science teams. These teams need to organize and maintain a body of information accumulated through scientific fieldwork, laboratory experimentation, and data analysis. The types of information stored by scientific teams is diverse, and includes scientific measurements, publication manuscripts, datasets, field site descriptions and photos, field sample records, electron microscope images, genetic sequences, equipment rosters, research proposals, etc. Various relationships among these types of information are represented within ScienceOrganizer, and they are used to link the information together within the repository. For example, a field sample can be: *collected-at* a field site; *collected-by* a person; *analyzed-by* an instrument; *imaged-under* a microscope; etc. We have selected two ScienceOrganizer applications to highlight in this section: EMERG and Mobile Agents.

The Early Microbial Ecosystems Research Group (EMERG) was an early adopter of ScienceOrganizer, and provided many of the requirements that drove its development. EMERG is an interdisciplinary team of over 35 biologists, chemists, and geologists, including both U.S. and international participants across eight institutions. Their

goal is to understand extreme environments that sustain life on earth and help characterize environments suitable to life beyond the planet. EMERG focuses on understanding the evolution of microbial communities functioning in algae mats located in high salinity or thermally extreme environments. As part of their research, they conduct field trips and collect mat samples in various remote locations, perform field analysis of the samples, and ship the results back to laboratories at their home institution. There, they perform experiments on the samples, grow cultures of the organisms in the mats, analyze data, and publish the results.

ScienceOrganizer was used across the team to store and interlink information products created at each stage of their research. This enabled the distributed team to work together and share information remotely. As a side benefit, the repository served as an organizational memory [15] that retained a record of previous work that could be referenced when planning subsequent scientific activities. As part of the collaboration with EMERG, we developed a capability that allows scientists to set up and initiate automated laboratory experiments on microbial mats from within ScienceOrganizer. The scientist defines an experiment within ScienceOrganizer by specifying its starting time and providing details of the experimental parameters to be used. A software agent is responsible for controlling internet-accessible laboratory hardware and initiating the experiment at the specified time. When the experiment is complete, the agent deposits experimental results back within ScienceOrganizer so they can be viewed by the scientist. This capability allows remote users to initiate experiments and view results from any location using ScienceOrganizer.

The second project, Mobile Agents [16], is a space mission simulation that uses mobile software agents to develop an understanding of how humans and robots will collaborate to accomplish tasks on the surface of other planets or moons. As part of the mission simulation, humans (acting as astronauts) and robots are deployed to a remote desert location, where they conduct a mock surface mission. In this context, ScienceOrganizer is used as a repository for information products generated during the mission, including photos, measurements, and voice notes, which are uploaded by autonomous software agents using the system's XML-based API. ScienceOrganizer also serves as a two-way communication medium between the mission team and a second team that simulates a set of earth-bound scientists. The science team views the contents of ScienceOrganizer to analyze the field data uploaded by the mission team. In response, the science team can suggest activities to the mission team by uploading recommended plans into ScienceOrganizer for execution by the mission team.

5.3 InvestigationOrganizer

When an accident involving NASA personnel or equipment occurs, NASA policy requires the creation of an accident investigation board to determine the cause(s) of the mishap and formulate recommendations to prevent future accidents. Information management, correlation, and analysis are integral activities performed by an accident investigation board. Their primary tasks are to collect and manage evidence, perform different types of analysis (e.g., chemical, structural, forensic) that generate derivative evidence, connect the evidence together to support or refute accident hypotheses, conduct failure analyses, come to a resolution on accident causal factors, and make

recommendations. The heterogeneous nature of the evidence in NASA accidents coupled with the complex nature of the relationships among evidence and hypotheses make the use of a system like SemanticOrganizer quite natural in this setting. NASA accident investigation teams typically are composed of engineers, scientists, and safety personnel from NASA's ten geographically distributed field centers across the country. Each team is composed of specialists with expertise pertinent to the accident. Distributed information sharing is an essential capability for accident investigation teams. Although the team may start out collocated, evidence gathering and analysis often take team members to different sites. With lengthy investigations, the logistics of centralizing personnel and information at one location are unworkable. Teams have relied on standard information-sharing technology in past investigations: email, phone, fax, and mail courier. From many perspectives – security, timeliness, persistence – these approaches are largely inadequate.

InvestigationOrganizer was developed in partnership with NASA engineers and mission assurance personnel to support the work of distributed NASA mishap investigation teams. The types of data stored by these teams include a wide variety of information, including descriptions and photos of physical evidence, schematics and descriptions of the failed system, witnesses interviews, design and operational readiness documents, engineering telemetry, operator logs, meeting notes, training records, hypothesized contributory accident factors, supporting and refuting evidence for those factors, etc. Various relationships among these types of information are represented within InvestigationOrganizer and serve to link information (e.g., as in Figure 1). For instance, a design document can: *describe* a physical system; be *authored-by* a contractor employee; *refute* a hypothesized accident factor; be *requested-from* a contracting organization; etc.

To date, InvestigationOrganizer has been used with four NASA mishap investigations that ranged in scope from minor localized investigations to major distributed investigations. The larger investigations included the loss of the Space Shuttle Columbia as well as the loss of the CONTOUR unmanned spacecraft which disappeared while escaping earth orbit.

Within the Columbia and CONTOUR investigations, InvestigationOrganizer was used to track information pertaining to almost every aspect of the investigation. The system also supported analysis of the data in terms of fault models and temporal event models that were built to understand the progression and causes of the accidents. Mishap investigators in these investigations went beyond the system's basic capabilities to support evidence collection and correlation; they used InvestigationOrganizer to explicitly record and share investigators' reasoning processes as the investigations proceeded. In the case of the Columbia, an added benefit to recording these processes was a preservation of the chain of evidence from hypotheses and theories to findings and recommendations. This chain of evidence is currently being used in NASA's efforts to return the Space Shuttles to flight, allowing engineers to trace the reasoning behind the conclusions reached by the investigation board.

6 Lessons Learned

Our experience deploying SemanticOrganizer across numerous domains, and working with a very diverse set of users, has given us a glimpse into the promise and the perils associated with semantic repository applications. In this section we discuss some of our key lessons learned.

6.1 Network-Structured Storage Models Present Challenges to Users

Despite the ubiquity of the Web, we found that people are not initially comfortable with using network structures for storing and retrieving information. Most information repositories use the familiar hierarchical structure of folders and subfolders to organize files. While networks structures have advantages, the notion of connecting information using multiple, non-hierarchical relationships was very disorienting to some users. Even with training, they would either fail to comprehend the network model or reject it as overly complex and unnecessary for their needs. In response to users' desire to organize information hierarchically, we introduced nested folder structures into our repository with limited success. Folders were typed and linked to their contents via a 'contains' relation. Users could create folders of people, photos, biological samples, etc. However, this model was unfamiliar to users expecting to place a set of mixed items in a folder without constraint. Our attempt to graft hierarchical structures onto networks left much room for improvement and we continue to seek better, more intuitive methods of combining these two models.

6.2 Need for both 'Loose' and 'Tight' Semantics

People have widely differing styles regarding the manner in which they wish to organize information. At one end of the spectrum are the meticulous organizers who strove to understand and use the full power of the semantic representations in our system. They would carefully weave the semantic network around their repository content and suggest precise revisions and extensions to the global ontology. They appreciated the increased descriptive power of a "tight" (i.e., more precise) semantics and didn't mind taking the additional time required to appropriately annotate and link the new material. At the other end of the spectrum are the casual organizers – users who simply wanted to add their document to the repository as quickly as possible. If their new material didn't align easily with the existing semantics, they became frustrated. They wanted "loose" semantics that would minimally cover their situation so they could quickly add and link their material, yet feel comfortable it was at least reasonably correct. SemanticOrganizer was designed with the meticulous organizers in mind and we had to relax our notion of what was semantically correct to accommodate the casual organizers. However, we found that in our attempt to craft compromises and simultaneously accommodate both styles of use, we sometimes failed to serve either group properly.

6.3 Principled Ontology Evolution is Difficult to Sustain

Because we often had a half dozen projects in active development, ontology sharing and evolution became much harder than expected. Our knowledge modelers understood the importance of reuse and initially, there was sufficient momentum to evolve common ontology components to meet the changing needs of projects. However, as the workload and schedule pressures increased, it became increasingly difficult to coordinate the necessary discussions and create consensus on how to evolve the ontology. In an effort to meet their individual project needs, modelers would simply start cloning portions of the ontology and then evolve them independently. Cloning serves immediate local project needs and offers the freedom to quickly make decisions and updates without seeking global consensus. Because our tools for merging classes or morphing instances into new classes were not well developed, modelers were also reluctant to take the time during slower periods to recreate a more globally coherent ontology. We expect this will continue to be a difficult problem to address.

6.4 Navigating a Large Semantic Network is Problematic

Typical projects in SemanticOrganizer contain more than 5000 informational nodes with 30,000 to 50,000 semantic connections between those nodes. A common user complaint with SemanticOrganizer is the difficulty of orienting themselves in the information space. The standard system interface (Figure 4) presents the details of a single node and a hyperlinked list that names all of its direct neighbors, organized by the semantic type of the link. This interface is convenient for editing the informational content of a node and linking it to new neighbors, but it does not help with non-local navigation. The degree of the node connectivity is bimodal with a small, but significant, percentage of the nodes being connected to many tens of nodes, while 30 to 40 percent of the nodes have 3 or fewer links. Imagine trying to explore a city having several massive central intersections where hundreds of streets meet. Most of these streets are narrow paths leading thru smaller intersections ending in a cul-de-sac. Interfaces that allow users to understand the overall topology of the space and that allow a smooth transition from a local to a global perspective are needed to understand how information is connected to items that are important to a user's task [17].

6.5 Automated Knowledge Acquisition is Critical

The original design concept for SemanticOrganizer was that teams would primarily manage their repository space manually, using the web interface to add links, enter new information, and upload artifacts such as documents or scientific measurements. But we quickly found that the task of adding information to the repository and linking to existing content can be time consuming and error prone when the volume of information is large or many people are involved. To address this need, SemanticOrganizer evolved to incorporate various forms of automated knowledge acquisition: an inference engine that uses rules to create links between items and maintain the semantic consistency of the repository; an API that allows software agents to add artifacts,

modify meta-knowledge, and create links; a Microsoft Office macro that give users the ability to upload information directly from an Office application; and an email processing system that incorporates user email directly into SemanticOrganizer. We now understand the importance of developing knowledge acquisition methods that allow users to seamlessly add new repository content as a by-product of their normal work practices, without imposing the burden of new tools or procedures.

7 Summary and Future Directions

Developing the SemanticOrganizer system has left us with a solid foundation of experience in developing practical semantic web applications. The application domains and users we've directly supported are extremely diverse and have ranged from a few highly specialized research scientists exploring evidence of microscopic signs of life on Mars, to retired generals and executives of major aerospace companies, leading the investigation into the tragic loss of Columbia. SemanticOrganizer represents a microcosm of the benefits and challenges that will become part of a broadly distributed and implemented semantic web vision of the future.

As an early large-scale semantic web application effort, many of our system's components were designed and build to accommodate our urgent engineering requirements, prior to recent semantic web standardization efforts. We are currently rearchitecting our system to standardize selected components and improve our interoperability with other emerging semantic web tools. This will include a transition into a web service compatible framework. In addition to refining our access control and personalization frameworks, we have also begun work on a number of new capabilities following from our lessons learned. In particular, we are working on new visualization techniques to provide users with an enhanced ability to understand and navigate the semantic repository. We are also building tools that will automatically analyze text from documents and produce semantic annotations that link the document to related items in SemanticOrganizer.

Acknowledgments

We gratefully acknowledge funding support by the NASA Intelligent Systems Project and by NASA Engineering for Complex Systems Program. This work would not have been successful without dedicated application partners in various scientific and engineering disciplines. Tina Panontin and James Williams provided invaluable guidance and direction on the application of SemanticOrganizer to accident investigation. They also took a leading role in the deployment of SemanticOrganizer for the Shuttle Columbia accident investigation, as well as other investigations. Brad Bebout provided essential long-term guidance and support for the application of SemanticOrganizer to astrobiology and life science domains. Maarten Sierhuis provided support for application to space mission simulation testbeds. Our sincere appreciation goes to our colleagues Sergey Yentus, Ling-Jen Chiang, Deepak Kulkarni, and David Nishikawa for their contributions to system development.

References

1. T. Berners-Lee, "A Roadmap to the Semantic Web," 1998, http://www.w3.org/ DesignIssues /Semantic.html.
2. R. M. Keller, "SemanticOrganizer Web Site," 2004, http://sciencedesk.arc.nasa.gov.
3. B. R. Gaines and D. Madigan, "Special Issue on Knowledge-based Hypermedia," *International Journal of Human-Computer Studies*, vol. 43, pp. 281-497, 1995.
4. D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C, 2004, http://www.w3.org/TR/rdf-schema/.
5. E. Friedman-Hill, "Jess: The rule engine for the Java platform," 2004, http://herzberg.ca.sandia.gov/jess/index.shtml.
6. G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, pp. 39-41, 1995.
7. D. C. Berrios and R. M. Keller, "Developing a Web-based User Interface for Semantic Information Retrieval," in *Proc. ISWC Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, N. Ashish and C. Goble, Eds. Sanibel Island, FL, 2003, pp. 65-70.
8. O. Corcho, A. Gomez-Perez, A. Lopez-Cima, V. Lopez-Garcia, and M. Suarez-Figueroa, "ODESeW. Automatic generation of knowledge portals for Intranets and Extranets," *The Semantic Web - ISWC 2003*, vol. LNCS 2870, pp. 802-817, 2003.
9. Y. Jin, S. Xu, S. Decker, and G. Wiederhold, "OntoWebber: a novel approach for managing data on the Web.," *International Conference on Data Engineering*, 2002.
10. N. Stojanovic, A. Maedche, S. Staab, R. Studer, and Y. Sure, "SEAL - a framework for developing semantic portals.," *Proceedings of the International Conference on Knowledge capture*, pp. 155-162, 2001.
11. P. Spyns, D. Oberle, R. Volz, J. Zheng, M. Jarrar, Y. Sure, R. Studer, and R. Meersman, "OntoWeb - a semantic Web community portal.," *Fourth International Conference on Practical Aspects of Knowledge Management*, 2002.
12. E. Bozsak, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, G. Stumme, Y. Sure, J. Tane, R. Volz, and V. Zacharias, "KAON-towards a large scale Semantic Web.," *Proceedings of EC-Web*, 2002.
13. "BrainEKP." Santa Monica, CA: TheBrain Technologies Corporation, 2004, http://www.thebrain.com.
14. A. P. Sheth and C. Ramakrishnan, "Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis," *IEEE Data Engineering Bulletin*, vol. 26, pp. 40-48, 2003.
15. R. Dieng-Kuntz and N. Matta, "Knowledge Management and Organizational Memories." Boston: Kluwer Academic Publishers, 2002.
16. M. Sierhuis, W. J. Clancey, C. Seah, J. P. Trimble, and M. H. Sims, "Modeling and Simulation for Mission Operations Work Systems Design," *Journal of Management Information Systems*, vol. 19, pp. 85-128, 2003.
17. V. Geroimenko and C. Chen, "Visualizing the Semantic Web: XML-based Internet and Information Visualization." London: Springer-Verlag, 2003.