

CS AKTive Space, or How We Learned to Stop Worrying and Love the Semantic Web

Nigel Shadbolt, Nicholas Gibbins, Hugh Glaser, Stephen Harris, and m.c. schraefel,
University of Southampton

The Advanced Knowledge Technologies project (www.aktors.org) is a six-year, £7.5-million effort in which five universities hope to operationalize the knowledge management mantra, “Getting the right content to the right place, at the right time, and in the right form.” From AKT’s outset in October 2000, it has been investigating how to

realize the potential of the extraordinary information repository that humankind is building—the World Wide Web.

A significant result from this research is CS AKTive Space, a Semantic Web application that won the 2003 Semantic Web Challenge. Submissions to the challenge had to demonstrate that the content they drew on was geographically distributed and comprised diverse ownership, syntactic and semantic heterogeneity, and real-world data. Submissions also had to assume an open world in which information is never complete, as well as use some formal description of the data’s meaning. CAS addresses each of these requirements to provide an integrated information overview of who’s doing what and where in UK university-based computer science research.

Additionally, CAS addresses another key Semantic Web issue—scale. Users need a way to rapidly explore a potentially vast information space to acquire an overview. Yet when examining that information to inform their decision, they must be able to focus on a specific area. For example, someone organizing a workshop in a particular discipline might want to identify the discipline’s important researchers (a vast information space) to determine whom to invite and then identify their locations (a specific area) to determine where to hold the workshop. CAS could help with such a task.

In developing the application, we engaged numerous core Semantic Web challenges: acquiring con-

tent, developing ontologies to mediate heterogeneous data sources, developing scalable RDF storage and query facilities, semantically directing interaction design, and facilitating knowledge-processing services over the harvested content.

CS AKTive Space

CAS provides multiple ways to look at and discover simple information and rich relations in computer science research. We chose this domain because

- We had a real interest in having such a set of services.
- It’s a domain we understand.
- It’s relatively accessible and easy to communicate as a domain.
- We could secure access to a wide range of content that wasn’t subject to industrial embargo.
- It presented real challenges in terms of scale and scope.

Furthermore, CAS addresses a scenario that originated as a real-world community request: a funding council’s desire to quickly obtain an overview of its domain from multiple perspectives. This requires bringing together data from heterogeneous sources and constructing methods to present possible relations in the data quickly and effectively.

For example, CAS gathers content on a continuous basis using a variety of methods including har-

*CS AKTive Space
addresses many core
Semantic Web
challenges to offer an
integrated information
overview of who’s
doing what and where
in UK computer
science research.*

```

SELECT ?uri, ?name
WHERE (?uri, <rdf:type>, <akt:Person>),
      (?uri, <akt:has-full-name>, ?name),
      (?uri, <akt:works-on>, ?project),
      (?project, <akt:has-url>, <http://aktors.org/>)
USING akt FOR <http://aktors.org/ontology#>

```

Figure 1. RDF Data Query Language (RDQL) query returning the identifying uniform resource identifier (URI) and names for all people working on the Advanced Knowledge Technologies project.

vesting and scraping of publicly available data from institutional Web sites,¹ bulk translation from existing databases, and direct submissions by partner organizations. In particular, it supports both regularly scheduled harvesting to identify and deal with changes to existing data sources, and on-demand harvesting in response to changing user requirements² or update notifications from component sources. Its information resources include Web sites of university departments and UK funding agencies, homepages of individuals, databases containing the results of national research assessment exercises, and various resources containing bibliographic and publication data (such as geographical gazetteers and UK Research Assessment Exercise submissions). Its data includes almost 2,000 computer science faculty (who are actively doing research), 24,000 research projects, thousands of papers, and hundreds of distinct research groups.

More importantly, CAS can query, explore, and organize this data in ways that are meaningful to the user. Where one user or group (such as a funding-council member or researcher) might be interested in seeing the relationship between funding, research area, and geographical region, another (a graduate student, perhaps) might be interested in identifying top AI researchers and obtaining their phone numbers. CAS can

- Browse topics and institutions for researchers
- Show the geographic range and extent of research topics
- Estimate top researchers (in terms of scholarly impact³ and cumulative research grant income) by topic and geographic region
- Calculate a researcher's *community of practice*⁴ (the researcher's coauthors and the projects in which they're involved, institutions with which they're affiliated, and topics in which they conduct research)
- Identify gaps in research coverage

So, users can formulate and see at a glance rich results such as these without having to string together large, complex queries.

To make content exploration, navigation, and appreciation direct and intuitive, CAS exploits a variety of visualizations and multidimensional representations.⁵ To this end, it integrates several knowledge services.

Services

As the World Wide Web benefits from its distributed nature in terms of scalability and robustness, we expect the same to be true of the Semantic Web. One way to achieve this distribution is to decompose Semantic Web systems into modules that provide specific competencies, an approach that's entirely in accord with the philosophy behind developments in the Web Services community.

The CAS system's core is a collection of Web services that communicate via HTTP and collaborate to provide the knowledge capabilities that the user interface requires. Currently, these services are explicitly configured and organized, and are referred to using hard-coded URLs in the relevant system components. We're moving from this to a system that dynamically binds the services using service discovery techniques, which will reduce brittleness and make the system less dependent on specific service instances.

CAS currently has five main service types.

3store

A central CAS component is the RDF Schema triplestore, which evaluates queries and performs simple inferences on the information the system uses. We use 3store, an RDF Schema triplestore server,⁶ developed in the AKT project to provide a scalable RDF retrieval and reasoning service. 3store can return query results within a few milliseconds; the user interface might generate dozens of queries for each user action, and interactive performance requires a swift response.

3store provides a Web Services interface to its query engine, which accepts RDF Data Query Language⁷ queries and returns XML documents containing the results. RDQL is an SQL-like language for performing subgraph pattern matches over an RDF graph, returning bindings for the variables specified in its SELECT clause. Figure 1 shows an example RDQL query.

Ontocopi

The community-of-practice service identifies implicit communities by finding sets of instances associated with a selected instance in a knowledge base. Through the user interface, a user can request more detailed information on an individual, including his or her community of practice. We generate this information using the Ontocopi⁸ service, also developed within the AKT project.

Ontocopi uses *ontological network analysis* to discover connections between the objects that the ontology only implicitly represents. For example, the tool can discover that two people have similar interaction patterns, work with similar people, attend the same conferences, and subscribe to the same journals.

Ontocopi is invoked via HTTP and uses the RDF Schema triplestore to perform its ONA process (speed isn't as critical as for the direct user interface interactions because Ontocopi calculates and displays the community-of-practice results asynchronously). It supports an HTTP interface, which takes an individual's uniform resource identifier, along with the temporal extent of the relations traversed and the maximum number of relations to be traversed out from the source individual, and returns an XML document. That document then contains the URIs of the related individuals and a ranking factor.

Geographic visualizer

This service provides a graphical representation of the geospatial information in the ontology (the locations of institutions of interest) and lets the user directly specify geographical constraints.

Scheduled harvesters

The harvesters extract information from Web sites, databases, spreadsheets, and other information sources, convert it into RDF form using an appropriate ontology, and assert it into the triplestore. The system invokes harvesters according to a predetermined schedule, from nightly to monthly, which then generate RDF files varying in size from a few hundred to several million triples.

Dynamic harvester

This service takes instances that are underpopulated in the knowledge base and produces more knowledge about them. It uses a set of natural language techniques to extract knowledge from Web sources and formats it according to CAS's ontology.

We use the Armadillo service,⁸ which uses RDQL queries to determine what knowledge is already asserted regarding the instance to be populated, and to help resolve referential integrity issues.⁹ This preexisting knowledge then informs natural language searches (over a variety of Web sources), which extract further knowledge and assert it into the triplestore. In this way, future requests for information about the instance in question will potentially return more information.

As with Ontocopi, Armadillo must be able to query an RDF store, but it also must be able to assert the new knowledge that it extracts. Because Armadillo takes approximately 15 minutes to perform information extraction, we describe it as an asynchronous service, which does not return a result. The success or failure notification wouldn't be timely enough to be of use to the user.

Harvesting

Owing to the wide variety of data sources we used, we had to develop individual services for each of our data sources that mediate and recast these sources in terms of our ontology, as outlined in the previous section. These services range from specialized database export scripts to XML transformation tools¹ trained to extract the required content from semistructured Web pages. Although these mediators are based on a common framework of code (which handles the rote work of database access, HTTP retrieval, RDF construction, and the more common patterns in our ontology, such as date and time expression), they each contain specialized capabilities tailored to the individual data sources' content and nature. Although such a mediator's bulk translation of instance data is straightforward, our use of these mediators has shown that mapping existing structured and semistructured data at the schema or ontology level isn't a task that we can effectively automate in all cases. Investing effort in building mediators for our common ontology is reflected in the consequent perceived value of the knowledge base to which they contribute, because it reduces the likelihood of errors when translating the data sources.

CAS requires that a range of content be available for the system to use. Some of this content already exists in suitable structured forms; other content doesn't. We adopt a pragmatic attitude: although the content we're gathering is the prime mover driving the interface, we should also tolerate inconsistencies. We use available data sources as best we can, antici-

pating that we'll be able to better use them in the future. Although this comes at a cost (there's an implicit commitment to future knowledge maintenance), such early exploitation of available content is necessary to initiate a community process that should be self-sustaining in the future, thus justifying our effort.

Similarly, we also expect data on the Web to eventually be marked up, either mechanically or by hand. However, in the medium-to-short term, suitable sources for our application domain won't be available. From the outset, we've been gathering the data we need, so that when the storage and processing technologies are built, the data will be there.

In gathering the data, we employ both push and pull models of knowledge acquisition.

The physical distribution of data is much less important than the semantic distribution of data brought about by using disparate ontologies for the same application domain.

The push model involves a data source (the publisher) choosing to express its data in terms of the ontology that CAS uses. The publisher is solely responsible for the translation, so the consumer can simply retrieve the translated knowledge base without any further effort required on his or her part. In comparison, the pull model requires that the consumer take a raw data source (which might be published against some other ontology or might exist only as a set of unannotated Web pages) and construct a knowledge base from it.

In some ways, the pull model is more advantageous because it gives the consumer more control over what information the resulting knowledge bases encode. Also, the consumer can more easily correct inconsistencies or adapt to changes in the underlying common ontology. However, this comes at a greater cost to the consumer, in both the knowledge lifecycle's acquisition phase (when a new data source is acquired) and the maintenance phase.

We use the pull model predominantly for large, comparatively static data sources (for example, the list of countries and administrative regions given by ISO3166, a gazetteer

maintained by the International Standards Organization). We also use it as an interim solution for high-value data sources of general interest to the community (for example, the Engineering and Physical Sciences Research Council's database of research funding). This "pump-primes" the system with sufficient data, encouraging other community members to push their local data sources to us and initiating a viral, rich-get-richer phenomenon. Eventually, we aim to encourage the owners of most of these pull-model sources to move to a push delivery model.

The information these mediators gather consists of 430 Mbytes of RDF/XML files, which are published on the hyphen.info Web site and contain approximately 10 million RDF triples describing 800,000 instances of people, places, publications, and other items of interest to the academic community.

The current development of knowledge services on the Semantic Web raises many issues not commonly encountered in existing knowledge-based systems. These issues pertain to the distribution of knowledge and the difficulty of agreeing on a conceptualization in a distributed environment that has no ultimate authority. For example, *coreference* arises when more than one URI refers to a given resource, causing problems when the system combines statements from different knowledge bases. We have three complementary approaches to address this problem with CAS. First, we support the simple social solution, where we use the emerging knowledge base as a gazetteer or name authority, so that new knowledge can be asserted with a common agreement on URIs. Second, we have heuristic methods that conservatively coalesce appropriate entities⁹ (using `owl:someAs` assertions). Third, we developed a coreference editor that builds on the second heuristic approach but allows user intervention.

Ontology development

Even in a distributed environment such as the Semantic Web, the physical distribution of data is much less important than the semantic distribution of data brought about by using disparate ontologies for the same application domain. We use a single common ontology to express the data that drives CAS. This AKT Reference Ontology mediates and guides the integration of the different data sources. When expressed in terms of our ontology, the RDF data that these sources obtain is made publicly available through the hyphen.info Web site.

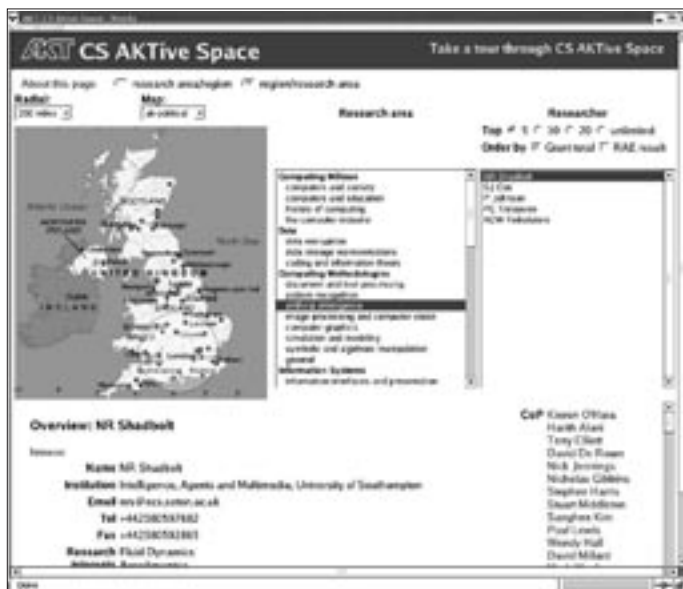


Figure 2. The CAS interface. The columns present a visual, directly manipulable representation of a complex query. Filters above the columns let users refine their queries further. Selecting an element in one column constrains the instances that will populate the following column. The interface's lower half presents an overview of the information associated with the currently selected instance.

We developed the AKT Reference Ontology over a six-month period, drawing heavily both on ontologies written previously by AKT partners and on other published ontologies.¹⁰ We took this approach because ontology development is costly, and careful reuse of existing work reduces duplicated effort. Also, our choice of a single ontology for the system reduces the amount of translation required. Information from external sources (which might have been expressed in other ontologies) still must be translated into the system ontology, creating a “walled garden” in which working is easier.

We chose the closed approach for pragmatic reasons. The alternative would have been to use heterogeneous ontologies throughout the system. This might have complicated the design of the system components unnecessarily, because each component must be able to either translate between each of the ontologies in use or invoke a separate translation service that mediates between the ontologies. Additionally, these ontologies might have mutual inconsistencies, raising the cost of deciding on appropriate mappings between

ontologies or ontology fragments. Our closed approach is also more practical for writing tools for information extraction against a single ontology and for writing tutorial materials from which third parties can learn to write their own extraction tools.

Our ontology choice was driven largely by our previous developments, but we hypothesized that we could construct an *mSpace* interface for an application domain given any suitably expressive ontology for that domain. (*mSpace* is an interaction model that represents high dimensional domains through a defined set of manipulations on the domain's dimensions.⁵) Consequently, the actual ontology choice was largely immaterial, and this has been borne out in our experiences of building CAS. Selecting elements to be displayed in CAS brings certain characteristics of the application domain to the foreground (people, publications, institutions, and so forth) in what is essentially an abstract ontology for that domain. What is required is that the system designer can extract these characteristics from our chosen ontology in sufficient detail, regardless of the manner in

which our ontology represents them.

The chief concern with ontology choice then had to do with the interface's performance. Overly expressive structures in baroque ontologies might still extract the information that the user interface requires but at an exaggerated cost (of executing queries that have been expressed in that ontology). Our ontology proved itself fit for its purpose, although some areas were more complex than needed for the specific task (notably, representations of dates and publications). However, we anticipated this as a likely consequence of using a largely task-neutral ontology in a more task-specific application.

Interaction

The CAS interface integrates the services we've described so far into a single application for exploring the UK computer science domain (see Figure 2).

The interface's top half features a set of columns. The labels on the columns represent a set of queries against the ontology. In Figure 2, the columns represent the categories Region, Research Area, and Researcher. The selection in the column to the left acts as a constraint on the results of the column to the right. When the user selects one (or more) of the instances in a column, the detail view beneath the columns provides further information about the selected instance.

The combination of instance context, provided by seeing a selected instance in the context of other associated instances, along with detail about that selected instance, provides users with a fast visual means to interrogate relations among data. The visual representation of these queries means that the user can quickly reformulate them with new data by selecting different instances in a column. The detail view, along with the instance selection context of the surrounding instances in the column view, also means that users don't have to leave the context of the CAS window to gain at least two kinds of information about a particular instance: breadth and depth. Context in the column view provides a persistent view of alternate instances that match the given criteria for the current set of data presented. The proximity of one instance to another might provide new information about that part of the domain, possibly sparking a new query to, for instance, explore in more detail how the two instances are related.

In the state Figure 2 shows, the user has resorted the column views from the default arrangement of Research Area, Region, and

Researcher to Region, Research Area, and Researcher. The Region column shows that the user selected a 200-mile reticule in the map view and dragged it over part of the map representing southern England. From the list of research areas appearing in the Research Area column, the user selected Artificial Intelligence. With the constraints on Researcher set as Top 5, determined by Grant Total, the system lists five people in AI with the greatest grant totals. The user then selected one of the researcher names.

In the detail view, that researcher is the current selection. The left two-thirds of the view lists information about the researcher, including his full name, contact information, URL, and list of significant papers. The Ontocopi service automatically generates in the remaining third of the view the researcher's community of practice. Finally, at the window's bottom (not shown in the figure), the user can select the Run Armadillo service button. As we described earlier, Armadillo will search for additional information about this researcher to supplement the information already in the triplestore. Browsing the source data for the selected researcher will give that data's provenance.

Also in the detail view area, we embedded a "browse" link. Selecting browse at any point takes you to the sources from the triplestore, which inform the results presented in the detail view (see Figure 3). The final state of the user interface shows both the context of discovery for the current selected information and a view of the detailed information available about the researcher.

CAS represents an integration of a variety of well-tested interaction attributes for information access: it supports focus and context¹¹ in its persistent contextual information (via the instance labels in the columns) and its detailed information of any selected instance. Using a kind of query preview,¹² CAS supports the exploration of the domain space through complex underlying queries represented by the simple relations expressed in the columns. This facilitates users' contextual exploration of the domain via rapid selections of instances within columns.

A critical CAS attribute is that it persistently represents the information's context through direct manipulation¹³ in a spatial layout via the multicolumn view. This kind of layout is in contrast to the usual temporal Web model in which clicking a link causes a new page to load, replacing the previous information with the new page. Such a temporal



Figure 3. Browsing the triplestore, showing the underlying content and its provenance.

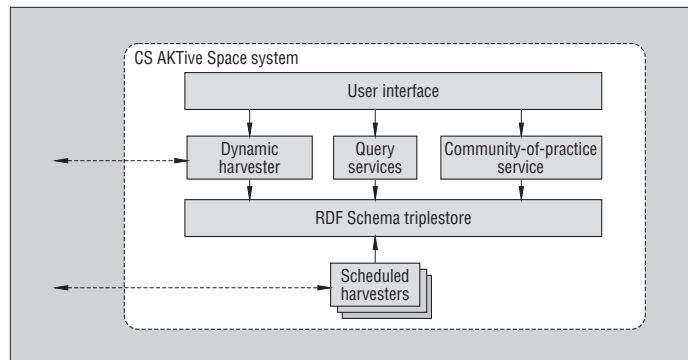


Figure 4. Component interactions in the CAS system.

approach places a higher cognitive load on the user to remember the relevant previous information. The spatial approach lets users focus on knowledge building with the information discovered rather than remembering which instance was next to another while focusing on the detail of a current selection. Similarly, the detail view provides specific information for the selected instance, enhancing the depth of knowledge about the instance beyond the label value provided in the column view.

We're pursuing numerous areas for future research. We're writing OWL Services¹⁴ descriptions for the services shown in Figure 4, so that future revisions of CAS can opportunistically discover and use new services in the system and adapt to the withdrawal of services. Also, AKTive Spaces will eventually work with multiple heterogeneous ontologies by coalescing them, mapping between them, or a combination of both.

Additionally, future iterations will better

The Authors



Nigel Shadbolt is a professor of artificial intelligence at the University of Southampton. His research interests include knowledge and Semantic Web technologies and biologically inspired computing. He received his PhD in artificial intelligence from the University of Edinburgh. He's editor in chief of *IEEE Intelligent Systems*, a fellow and a vice president of the British Computer Society, and a chartered member of the British Psychological Society. Contact him at Electronics and Computer Science, Univ. of Southampton, Southampton SO17 1BJ, UK; nrs@ecs.soton.ac.uk.



Nicholas Gibbins is a research fellow in the Intelligence, Agents, Multimedia research group of the School of Electronics and Computer Science at the University of Southampton. His research interests include applying Semantic Web technologies to organizational knowledge management and designing scalable infrastructures for multiagent systems, open hypermedia, and the Semantic Web. He received his PhD in computer science from the University of Southampton. Contact him at Electronics and Computer Science, Univ. of Southampton, Southampton, SO17 1BJ, UK; nmg@ecs.soton.ac.uk; www.ecs.soton.ac.uk/~nmg.



Hugh Glaser is a reader in computer science in the School of Electronics and Computer Science at the University of Southampton. His research interests include programming languages—particularly visual and declarative—parallel and distributed systems, and software engineering. He is currently applying these principles to the Semantic Web and related technologies. He graduated in computer science from the University of London. Contact him at Electronics and Computer Science, Univ. of Southampton, Southampton SO17 1BJ, UK; hg@ecs.soton.ac.uk.



Stephen Harris is a researcher in the Intelligence, Agents, Multimedia research group of the School of Electronics and Computer Science at the University of Southampton. His research interests are Semantic Web technologies, including scalability issues, hypertext, and metadata. He received his degree in computer science from Southampton University. Contact him at Electronics and Computer Science, Univ. of Southampton, Southampton, SO17 1BJ, UK; swh@ecs.soton.ac.uk; www.ecs.soton.ac.uk/~swh.



m.c. schraefel is a senior lecturer in the School of Electronics and Computer Science at the University of Southampton. Her research interests are in human-computer interaction—specifically, accessing and exploring large information spaces, both on and off the desktop. Her latest work has been in mSpace and in Making Tea (design methods to tackle loosely structured, highly expert, longitudinal tasks such as science experiments). Contact her at mc-cas@ecs.soton.ac.uk.

use the mSpace interaction model, which supports operations that enable interaction customization. One attribute is dimensional sorting, which lets users re-sort columns and hierarchies. The model also lets users replace one dimension with another (swap out Researcher for Project, for instance) or add new dimensions (add Project or Awards) or remove dimensions (look at only Region and Researcher). In each case, these affordances let users configure the space to support their interests in the domain. CAS has also implemented only dimensional sorting. Future

AKTive Space applications will include more of these model features.

Likewise, although CAS is a successful demonstration of an integrated Semantic Web application, our goal is to generalize the application so that AKTive Spaces can be deployed for any semantically defined domain. Part of the requirements to support such a generalized application is to automate the deployment of the interaction against the given ontologies as much as possible. To support this automation, we've been investigating the formalization of the mSpace interaction model.

We've also identified other Semantic Web challenges: What are the best ways to sustain an acquisition and harvesting activity? How should we model harvested content? How should we cope with numerous duplicate items that must be recognized as referring to the same objects or referents? The inferential service's ability to scale as more content becomes available will prove an issue sooner or later. How should we present the content to directly discern inherent patterns and trends? How trustworthy is the content's provenance and accuracy? How should we maintain and sustain this information as a social and community exercise?

The way in which further services might enhance the user interaction is still open; using Armadillo and Ontocopi illustrates how services of particular types can work with AKTive Spaces, but we can expect other services to require other interactions with AKTive Space. Finally, we're now working on generalizations to AKTive Space, so that the system can work in a domain-independent fashion. ■

Acknowledgments

We're pleased to acknowledge funding from the UK Engineering and Physical Sciences Research Council under grant GR/N15764/01 and Advanced Knowledge Technologies. We also thank the other partners on AKT, the Universities of Aberdeen, Edinburgh, and Sheffield, and the Open University, for their support.

References

1. T. Leonard and H. Glaser, "Large-Scale Acquisition and Maintenance from the Web without Source Access," *Proc. Workshop 4, Knowledge Markup and Semantic Annotation (K-CAP 2001)*, ACM Press, 2001, pp. 97–101.
2. F. Ciravegna et al., "Integrating Information to Bootstrap Information Extraction from Web Sites," *Proc. Workshop Information Integration from Web Sites*, 2003, www.isi.edu/info-agents/workshops/ijcai03/papers/ciravegna2.pdf.
3. S. Kampa, *Who Are the Experts? E-scholars in the Semantic Web*, doctoral dissertation, Dept. of Electronics and Computer Science, Univ. of Southampton, 2002.
4. H. Alani et al., "Ontocopi: Using Ontology-Based Network Analysis to Identify Com-

- munities of Practice." *IEEE Intelligent Systems*, vol. 18, no. 2, 2003, pp. 18–25.
5. m.c. schraefel, M. Karam, and S. Zhao, "mSpace: Interaction Design for User-Determined, Adaptable Domain Exploration in Hypermedia," *Proc. Workshop Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2003)*, 2003, pp. 217–235.
 6. S. Harris and N. Gibbins, "3store: Efficient Bulk RDF Storage," *Proc. 1st Int'l Workshop Practical and Scalable Semantic Systems (PSSS 03)*, 2003, pp. 1–20; <http://eprints.aktors.org/archive/00000273>.
 7. A. Seaborne, "RDQL—A Query Language for RDF," World Wide Web Consortium, 2004, www.w3.org/Submission/2004/SUBM-RDQL-20040109.
 8. A. Dingli, F. Ciravegna, and Y. Wilks, "Automatic Semantic Annotation Using Unsupervised Information Extraction and Integration," *Proc. SemAnnot 2003 Workshop*, 2003, pp. 1–8.
 9. H. Alani et al., "Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web," *Proc. 13th Int'l Conf. Knowledge Eng. and Knowledge Management (EKAW 02)*, Springer-Verlag, 2002, pp. 317–334.
 10. I. Niles and A. Pease, "Towards a Standard Upper Ontology," *Proc. 2nd Int'l Conf. Formal Ontology in Information Systems (FOIS 2001)*, ACM Press, 2001, pp. 2–9.
 11. D. Schaffer et al., "Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods," *ACM Trans. Computer-Human Interaction*, vol. 3, no. 2, 1996, pp. 162–188.
 12. C. Plaisant et al., "Interface and Data Architecture for Query Preview in Networked Information Systems," *ACM Trans. Information Systems*, vol. 17, no. 3, 1999, pp. 320–341.
 13. R. Chimera and B. Shneiderman, "An Exploratory Evaluation of Three Interfaces for Browsing Large Hierarchical Tables of Contents," *ACM Trans. Information Systems*, vol. 12, no. 4, 1994, pp. 383–406.
 14. *DAML-S: Semantic Markup for Web Services*, tech. report, DAML Services Coalition, May 2003, www.daml.org/services/daml-s/0.9/daml-s.html.