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

***Abstract*— The mission of the Advanced Knowledge Technologies (AKT) project is to investigate how to operationalise the knowledge management mantra of "getting the right content to the right place at the right time and in the right form." A significant result of the first three years of this six year project is CS AKTive Space (CAS), a Semantic Web application that won the 2003 Semantic Web Challenge. The challenge criteria included having to use geographically distributed, real world data that would have to be used in a context distinct from which that original data had been designed to serve. CAS is an application that, in meeting this criteria, seeks to provide the experience of an integrated information overview that allows a user to determine quickly who is doing what where in computer science research in the UK. In developing the application we have engaged a number of core Semantic Web challenges: content acquisition, ontology development to mediate heterogeneous data sources, scalable RDF storage and query facilities, and semantically directed interaction design. From our work on CAS we have begun to look at how the approaches for CAS can be generalised for the deployment of AKTive Space applications, dynamically generated from an ontology and set of services.**

***Index Terms*— Semantic Web; user interfaces**

I. Introduction

The Advanced Knowledge Technologies (AKT) project is a six-year, 7.5 million effort in which five universities are trying to understanding how to operationalise the knowledge management mantra of "getting the right content to the right place, at the right time and in the right form". From its outset in October 2000, AKT has been investigating how to realise the potential of the extraordinary information repository that humankind is building – the huge amounts of content held on the World Wide Web.

A significant result from this research is CS AKTive Space (CAS), a Semantic Web application that won the 2003 Semantic Web Challenge. Submissions to the challenge had to demonstrate that the content they drew on should have a necessary set of features: geographically distributed, diverse ownerships, syntactic and semantic heterogeneity, and should consist of real world data. A second requirement was that the applications assume an open world in which information can never be declared as complete. A final requirement was that the applications should use some formal description of the meaning of the data.

CAS is an application that seeks to provide the experience of an integrated information overview that allows a user to quickly determine who is doing what and where in computer science research within the UK, what are the community of practices of individual researchers, what impact is the work having. It harvests and has published to it a range of heterogeneous information from around the UK relating to academic computer science research. This collection is performed on a continual 24/7 basis. The information resources used include content scraped from the web sites of University Departments and the home pages of individuals, the web sites of UK funding agencies, databases containing the results of national research assessment exercises, and a variety of resources containing bibliographic and publication data.

The application comprises a number of core capabilities: content acquisition methods, an ontology to mediate the various heterogeneous content acquired, a scalable RDF storage and query facility, a semantically directed user interaction design, and a variety of knowledge processing services over the harvested content. In this paper we present an overview of the components of the CAS application. We discuss how these components could be generalised, the aim being to generate AKTive Spaces dynamically from an ontology and set of services.

II. Context

CS AKTive Space addresses a scenario that originated in a real-world community request: the desire for a funding council to be able to get a fast overview of the council's domain from multiple perspectives. This requires bringing together data from heterogeneous sources and constructing methods to present the possible relations in the data to a user quickly and effectively. We have found that our original scenario applies equally to any stakeholder of the domain. While a funding body may wish to know, for instance, how funding in an area is distributed geographically, a researcher may also wish to know this. Similarly, graduate students may wish to understand who is in the community of practice for the top researchers in their area.

CS AKTive Space attempts to provide an overview of current UK University based research in Computer Science by providing multiple ways to look at and discover simple information and rich relations within this domain. It facilitates querying, exploring and organising information in ways that are meaningful to the users: where one user or group may be interested in seeing the relationship between funding, research area and geographical region, another may be interested in who the top researchers are in AI and what their phone numbers are. With CS AKTive Space, people can formulate and see at a glance rich results like these, without having to string together large, complex queries.

To this end, CS AKTive Space provides services such as browsing topics and institutions for researchers, it can show

the geographic range and extent of research topics, provides an estimation of 'top' researchers in a topic and by geographic region, and is able to calculate a researcher's Community of Practice. We chose the CS domain for a number of reasons; (i) we had a real interest in having such a set of services, (ii) it is a domain that we understand, (iii) it is relatively accessible and easy to communicate as a domain, (iv) we were able to secure access to a wide range of content that was not subject to industrial embargo, (v) it presented real challenges of scale and scope.

Because of its rich, easily manipulatable representation of the CS domain, CS AKTive Space supports the exploration of patterns and implications inherent in the domain content. CS AKTive Space allows all stakeholders in the CS domain, from funding bodies to researchers, to explore their space for associations and opportunities that were previously either unavailable or too cumbersome to attempt to discover.

A critical component of CS AKTive Space is the interaction design and the affordances it provides for information representation and manipulation. The CAS interaction has been informed by mSpace [1], an interaction model designed to represent high dimensional domains through a defined set of manipulations on the dimensions of the domain, which we describe further in Section III-D. With the CAS work, we have begun to formalise the mSpace model; this formalism takes us closer to a framework for automating the deployment of mSpace interfaces for generalised AKTive Space applications.

In the following sections, we overview the attributes of the CS AKTive Space application. We conclude with a discussion of what a generic AKTive Space application would be, based on our experiences with CAS, and look at the challenges for automating the deployment of AKTive Spaces.

III. System Overview

CS AKTive Space exploits a variety of visualisations and multi dimensional representations that are designed to make content exploration, navigation and appreciation direct and intuitive [1], and integrates a number of knowledge services to this end. The services supported by the application include investigating communities of practice [2] and a researcher's prominence within their field, considered both in terms of their scholarly impact [3] and also of their cumulative research grant income.

To be convincing, a Semantic Web application also requires data on a large scale over which these services can operate. In the long term, we expect the data on the Web to be marked up, either mechanically or by hand. In the medium to short term, however, suitable sources for our application domain were not available. From the outset, we devoted effort to gathering the data we needed, so that when the storage and processing technologies had been built, the data would be there for them to use.

CS AKTive Space exploits a wide range of semantically heterogeneous and distributed content relating to Computer Science research in the UK. For example, there are almost 2000 research active Computer Science faculty, there are 24,000 research projects represented, many thousands of papers, and hundreds of distinct research groups. These entities

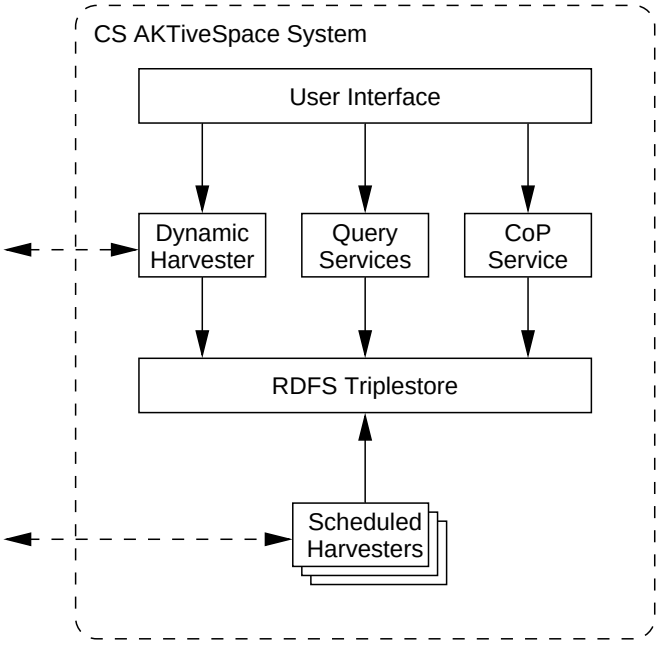

Fig. 1. Component interactions in the CS AKTive Space system

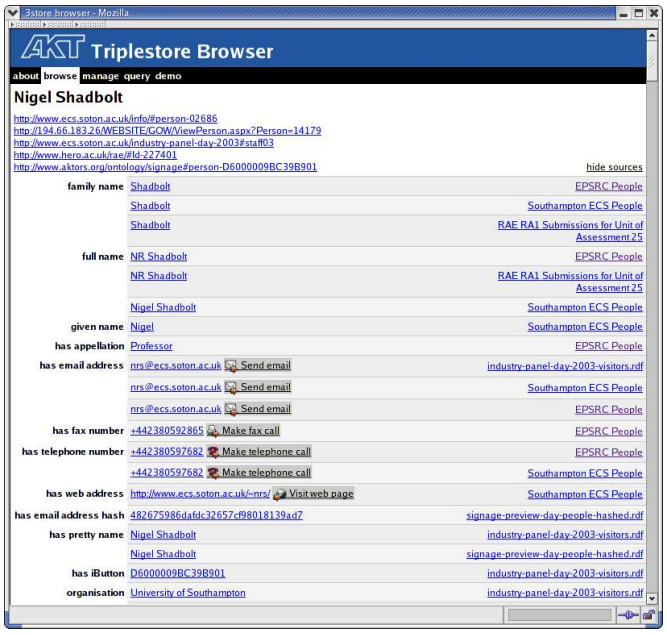

Fig. 2. Drilling down with the triplestore browser

are described by a number of existing sources, such as institutional information systems (university web sites, research council databases), bibliographic services and other third party data sets (geographical gazetteers, UK Research Assessment Exercise submissions).

This content is gathered on a continuous basis using a variety of methods including harvesting and scraping of publicly available data from institutional web sites [4], bulk translation from existing databases, and direct submissions by partner organisations, as well as other models for content acquisition. In particular, we support both regularly scheduled harvesting to identify and deal with changes to existing data

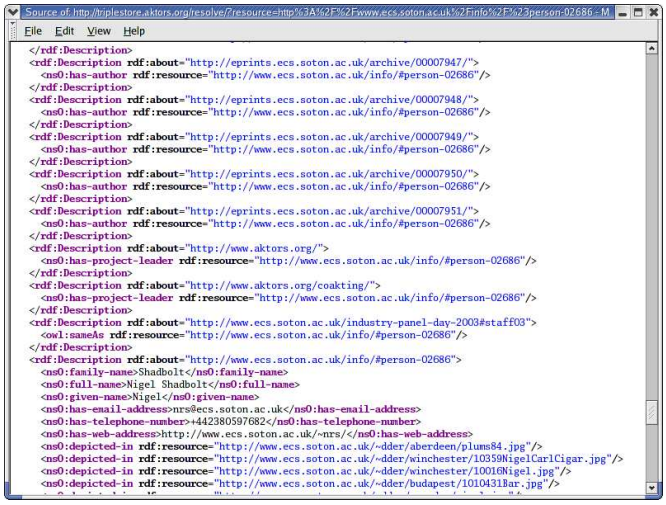

Fig. 3. Sample RDF information source

sources, and on-demand harvesting in response to changing user requirements [5] or update notifications from component sources.

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 the use of disparate ontologies for the same application domain. For the purposes of this application we use a single common ontology to express the data which drives the CS AKTive Space. We use this common ontology, the AKT Reference Ontology [6], to mediate and guide the integration of the different data sources. When expressed in terms of our ontology, the RDF data obtained from these sources is made publicly available through the `hyphen.info` web site.

The services which comprise CS AKTive Space rely on extensive inference across heterogeneous resources. For example, the notion of a community of practice is calculated from a given researcher's coauthors, the projects that they are involved with, the institutions with which they are affiliated and the topics in which they conduct research. We aim to provide a content space in which a user can rapidly get a Gestalt of who is doing what and where, what are the significant areas of effort both in terms of topic and institutional location, what of this work is having an impact or influencing others and where are the gaps in research coverage.

A. 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 in which this distribution may be achieved is by decomposing Semantic Web systems into modules which provide specific competencies, an approach which is entirely in accord with the philosophy behind the developments within the Web Services community.

The core of the CS AKTive Space system is a collection of Web Services that communicate via HTTP and collaborate to provide the knowledge capabilities required by the user interface. At present, these services are explicitly configured

```
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#>
```

Fig. 4. RDQL query returning the identifying URI and names for all people working on the AKT project

and organised, and are referred to using hard-coded URLs in the relevant system components. We are moving from this to a system in which the services are bound dynamically using service discovery techniques, which will have the effects of reducing brittleness and making the system less dependent on specific service instances.

CS AKTive Space currently comprises five main services or service types, as follows.

1) 3store: A central component of CS AKTive Space is the RDFS triplestore which is used to evaluate queries and perform simple inferences on the information used by the system. We use 3store [7], an RDFS triplestore server which was developed within the AKT project [8] in order to provide a scalable RDF retrieval and reasoning service. CS AKTive Space needs a triplestore which can return the results of a query within a few milliseconds: the user interface may 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 RDQL [9] queries and returns XML documents containing the results. RDQL is an SQL-like language for performing sub-graph pattern matches over the ground and inferred RDF graph, returning bindings for the variables specified in its SELECT clause. An example RDQL query is shown in Figure 4.

2) Ontocopi: The Community of Practice [2] service identifies implicit communities by finding sets of instances associated with a selected instance in a knowledge base. Through the user interface, a user may request more detailed information on an individual, including their community of practice. We generate the community of practice using the Ontocopi [2] service, also developed within the AKT Project. Ontocopi uses Ontological Network Analysis (ONA) to discover connections between the objects that the ontology only implicitly represents. For example, the tool can discover that two people have similar patterns of interaction, work with similar people, go to the same conferences, and subscribe to the same journals.

Ontocopi uses the RDFS triplestore to perform its ONA process (speed is not as critical as for the direct user interface interactions because the community of practice results are calculated and displayed asynchronously), and is invoked via HTTP. It supports an HTTP interface which takes an individual's URI, 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 containing the URIs of the related individuals and a ranking factor.

3) Geographic Visualiser: This service provides a graphical representation of the geospatial information within the ontol-

ogy (the locations of institutions of interest) and permits the user to directly specify geographical constraints.

4) Scheduled Harvesters: The harvesters extract information from websites, databases, spreadsheets and other information sources, convert it into RDF form using an appropriate ontology, and assert it into the triplestore. The harvesters are invoked according to a predetermined schedule, from nightly to monthly, and produce amounts of RDF varying from a few hundred to several million triples.

5) Dynamic Harvester: The dynamic harvester takes instances that are underpopulated in the knowledge-base and produces more knowledge about that instance. It uses a set of natural language techniques to extract knowledge from web sources and formats it according to the ontology used by CS AKTive Space.

We use the Armadillo [10] service, which uses RDQL queries to determine what knowledge is already asserted regarding the instance to be populated, and to help resolve referential integrity issues [11]. This pre-existing knowledge is then used to inform natural language searches (over a variety of web sources) that 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 needs to be able to query an RDF store, but it also needs to be able to assert the new knowledge that it extracts. Since Armadillo takes in the region of fifteen minutes to perform its information extraction process, we describe it as an asynchronous service which does not return a result; the success or failure notification would not be timely enough to be of use to the user.

B. Harvesting

Due to the wide variety of the data sources that we use, we have found it necessary to invest a degree of effort in developing 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 specialised database export scripts, to XML transformation tools [4] that have been trained to extract the required content from semi-structured 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 specialised capabilities that are tailored to the content and nature of the individual data sources. While the bulk translation of instance data by such a mediator is straightforward, our use of these mediators has shown that the mapping of existing structured and semi-structured data at the schema/ontology level is not a task that can be effectively automated in all cases; the investment of effort in building mediators for our common ontology is reflected in the consequent perceived value of the knowledge base to which they contribute.

The CS AKTive Space application requires that a range of content be available for use by the system. As it stands, some of this content already exists in suitable structured forms, while others do not. We adopt a pragmatic attitude that reflects the fact that although the content that we are gathering is the prime mover that drives the interface, we should also be tolerant of inconsistencies in that content. We make the immediate best use of the available data sources, perhaps in an imperfect fashion, while anticipating that we will be able to make better use of them in future. Although this comes at a cost (there is 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 and so justify the investment of effort.

We employ both push and pull models of knowledge acquisition, where push and pull refer primarily to whether the publisher or consumer are responsible for translating the data into a form which is suitable for the consumer. The push model involves a data source (the publisher) choosing to express its data in terms of the ontology used by the CS AKTive Space. The publisher is solely responsible for the translation, so the consumer may simply retrieve the translated knowledge base without any further effort required on its part. In comparison, the pull model requires that the consumer takes a raw data source (which may be published against some other ontology, or which may only exist as a set of unannotated webpages) and construct a knowledge base from that data source.

In some ways, the pull model has advantages over the push model, in that the consumer has a much greater level of control over what information is encoded within the resulting knowledge bases and is better placed to be able to correct inconsistencies or to adapt to changes in the underlying common ontology, but this comes at the expense of a greater cost to the consumer, both in the acquisition phase of the knowledge lifecycle (when a new data source is acquired), but particularly in 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), and as an interim solution for high-value data sources that are of general interest to the community (for example, the Engineering and Physical Sciences Research Council's database of research funding) as a means to 'pump-prime' the system with sufficient data to encourage other members of the community to participate by offering to push their local data sources to us, and so initiate a viral, rich-get-richer phenomenon. In the longer term, we aim to encourage the owners of the majority of these pull model sources to move to a push model of delivery.

The information gathered by these mediators consists of 430MB of RDF/XML files, which are published on the `hyphen.info` website and contain around 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 a number of issues which are not commonly encountered in existing knowledge based systems, and which pertain to the distribution of knowledge and the difficulty of obtaining agreement on a conceptualisation in a distributed environment when there is no ultimate authority. One such issue is that of coreference, which arises when more than one Uniform Resource Identifier is used to refer to a

given resource, and which causes particular problems when statements from different knowledge bases are to be combined. We have three complementary approaches to this problem for CS AKTive Space. Firstly, we support the simple social solution, where we allow the emerging knowledge base to be used as a gazetteer or name authority, so that new knowledge can be asserted with a common agreement on URIs. Secondly, we have heuristic methods that conservatively coalesce appropriate entities [11] (using `owl:sameAs` assertions). Thirdly, we have developed a coreference editor that builds on the second heuristic approach, but allows user intervention.

C. Ontology Development

CS AKTive Space uses the AKT Reference Ontology in order to mediate the disparate content that has been generated by the harvesting services. This ontology was developed within the AKT project over a six month period, and draws heavily both on ontologies written previously by AKT partners, and on other published ontologies [12]. The reasons for this approach are twofold: ontology development is a costly exercise, and careful reuse of existing work stands to provide benefits by reducing duplicated effort. Secondly, our choice of a single ontology for the system reduces the amount of translation required. Information from external sources (which may have been expressed in other ontologies) still needs to be translated into the system ontology, creating a 'walled garden' in which it is easier to work.

This closed approach was taken for pragmatic reasons. The alternative, being an open approach that might figuratively be described as a meadow in which many ontologies bloom, would have been to use a number of heterogeneous ontologies throughout the system. This approach runs the risk of complicating the design of the system components unnecessarily, because each of the components must either be able to translate between each of the ontologies in use, or be able to invoke a separate translation service that mediates between the ontologies. In addition, these ontologies may have mutual inconsistencies, which raises the cost of deciding upon appropriate mappings between ontologies or ontology fragments.

Our closed approach to ontology design has the additional benefit that it is more practical to write tools for information extraction against a single ontology, and much easier to write tutorial materials from which third parties can learn to write their own extraction tools.

Our choice of an ontology for the AKTive Space was driven largely by our previous developments, but we hypothesised that we would be able to construct an mSpace interface for an application domain given any suitably expressive ontology for that domain, and that the actual choice of ontology was largely immaterial. This has been borne out in our experiences of building CS AKTive Space. The selection of elements to be displayed in CS AKTive Space brings certain characteristics of the application domain to the foreground (people, publications, institutions, etc) in what is essentially a very abstract ontology for that domain. What is required is that these characteristics can be extracted from our chosen ontology in sufficient detail, regardless of the manner in which they are actually represented in our ontology.

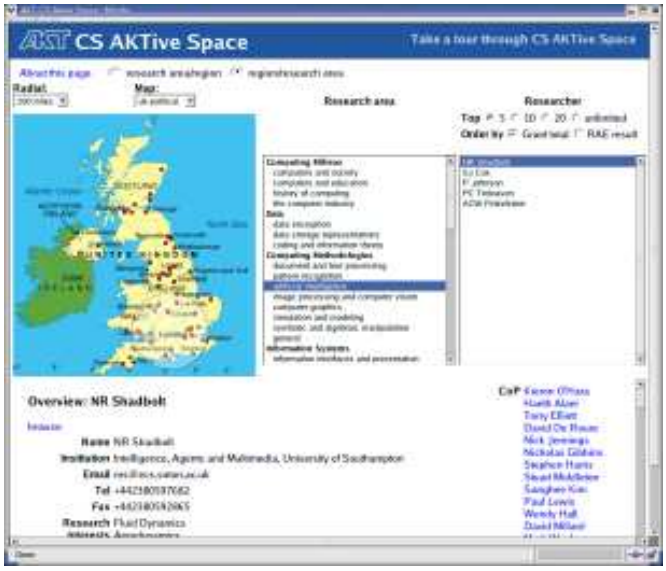

Fig. 5. CS AKTive Space

The chief concern with ontology choice is that of the resulting performance of the CS AKTive Space interface. Overly-expressive structures in baroque ontologies may still allow the extraction of the information required by the user interface, but at an exaggerated cost (being the cost of executing queries which have been expressed in that ontology). Our ontology showed itself to be fit for purpose, although some areas were more complex than they needed to be for the specific task performed by the user interface (notable areas were the representations of dates and of publications). However, this was anticipated as a likely consequence of using a largely task-neutral ontology in a more task-specific application.

D. Interaction

The goal of the Semantic Web is the representation of knowledge using Web technologies in order to let machines communicate with other machines in a semantically rich manner. Ultimately, that knowledge must be communicable back to the human beings that wish to make use of the system. To that end, the development of appropriate user interfaces for Semantic Web systems is an important area for Semantic Web research. CAS represents one approach for integrating a variety of services into a coherent application experience which supports the exploration of a domain from a variety of perspectives: the information can be re-arranged to suit the focus of users' queries.

While the interface hides the complexity of the services and queries in the application, it also allows users to go "beneath the hood" at any point to browse the sources of the provenance for any entity of interest presented in the UI. Thus, we provide multiple ways for users to engage the space to build up actionable knowledge. In order to contextualise the above interaction description, we present a brief walk-through of the interaction pictured in Figure 5.

The top half of the CS AKTive Space user interface (UI) features a set of columns. The labels on the columns represent a set of queries against the ontology. In the example given, the columns represent $< Region \mid Research\ Area \mid Researcher >$. The selection in the column to the left acts as a constraint on the results of the column to the right. When the user makes a selection of 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 representation of these queries visually means that they can be reformulated with new data quickly, simply by selecting different instances within a column. The detail view along with the instance selection context of the surrounding instances in the column view also means that users do not have to leave the context of the CS AKTive Space window in order 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 which match the given criteria for the current set of data presented. The proximity of one instance to another may provide new information about that part of the domain which might spark a new query in itself, to explore in more detail, for instance, how the two instances are related.

In the state pictured, the user has re-sorted the column views from the default arrangement of $< Research\ Area \mid Region \mid Researcher >$ to $< Region \mid Research\ Area \mid Researcher >$. The Region column shows that a 200 mile reticule has been selected in the map view, and the reticule has been dragged over part of the map representing southern England. From the list of research areas that shows up in the Research Area column, artificial intelligence has been selected. With the constraints on Researcher set as Top 5, determined by Grant Total, the list of 5 people in AI with the greatest grant totals are represented. One of the researcher names has then been selected. The Detail View shows that that researcher is the current selection: it is populated with information about the researcher, including full name, contact information, URL and list of significant papers on the left two-thirds of the view. In the right remaining third, the list of the researcher's community of practice is generated automatically by the Ontocopi service described in Section III-A.2. Finally, at the bottom of the window the user can invoke the Armadillo service by the "Run Armadillo" service button. As described in Section III-A.5, Armadillo will search for additional information about this researcher to supplement the information already present in the triplestore. Browsing the source data for the selected researcher will give the provenance of that data.

Within the Detail View area as well, we have embedded a "Browse" link (see Figure 5). By selecting browse at any point, one is taken to the sources from the triplestore which inform the results presented in the detail view, as shown in Figure 2. The final state of the UI shows both the context of discovery for the current selected information, as well as 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 + context [13] 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 [14] 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 attribute of CAS is that context of information is represented persistently via direct manipulation [15] 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. In such a temporal approach, there is a higher cognitive load on the user to remember the relevant previous information. The spatial approach provides persistence of information which means that users can focus their energies on knowledge building with the information discovered, rather than on splitting their attention, and hence increasing cognitive load, between 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.

Because the Semantic Web represents new possibilities for richer interaction with information than what the Web currently supports, our goal with the CAS interaction is to support interactions which support these new capabilities.

CAS therefore provides ways to engage information that go beyond and complement the Web paradigm of the keyword search box and the consequent clicking through resulting lists of links. Keyword search implies that seekers already have some specific queries they want addressed; they have a criteria to be matched. With CAS we wanted to support a user whose goals may also be less specific - users who rather than saying *"I want to know what Prof X's address is"* may want to ask *"what research is going on in this area of the country (close to where I'll be visiting); who is working in my research area there whose work I might not know?"*.

To support this kind of *domain exploration* we wanted to provide users with contexts they could explore within a domain. In such a domain representation, it would be easy to move from seeing who is working in a given research area in a particular location, to seeing who their community of practice is, to what some of any of those researchers' papers are, back to looking at someone or something else again, all so that the person can see this information in context. In other words, the interaction design would help support the changing interests or questions of a given user exploring within a domain. Our interest was to help provide application support for human-directed knowledge building. Bloom's Taxonomy of Learning [16] (see Table III-D) helps capture this.

Where some Semantic Web services support the lowest level

1) Evaluation: the ability to judge the worth of something
2) Synthesis: the ability to combine separate elements into a whole
3) Analysis: the ability to break a problem into its constituent parts and establish the relationships between each one
4) Application: the ability to apply rephrased knowledge in novel situation
5) Manipulation: the ability to rephrase or paraphrase knowledge
6) Knowledge: that which can be recalled

TABLE I
BLOOM'S TAXONOMY OF LEARNING

of the taxonomy – they present data which can be selected and recalled – the CAS application provides facilities to engage and support each of these phases of this knowledge building paradigm directly. By presenting knowledge in context, the interface facilitates multiple ways to see information, to represent or rephrase it in multiple contexts. By being able to change attributes of any selection within the domain contexts, one can see the role played by any selection on the end result. This comparison between states assists a person in making a valuation about a particular concept or proposal related to the information.

One could argue that an individual may progress through the same stages of Bloom's learning model by selecting, processing and assimilating the data in a list of links retrieved by a search engine. This certainly may be so. The idea of the CAS interaction, however, is to support these cognitive, knowledge building processes actively (or aktively). Robertson et al [17] have demonstrated the effectiveness for users' exploration and understanding of a domain of such such real time manipulation, in a similar way to that in which spread sheets support multiple ways to visualise and manipulate data for running what-if scenarios to determine the best course of action. The CAS interface likewise supports knowledge building through active, real-time exploration of the data by supporting users' abilities to reorganise the information to suit their focus of interest, and to observe the effects directly of such rearrangement. Indeed, it is the demonstration of such context-supporting interactions that has captured the interest of external agencies and organisations in taking up AKTive Spaces.

E. AKTive Space interfaces and the Semantic Web

While CAS has been a successful demonstration of an integrated Semantic Web application, our goal is to generalise the application so that AKTive Spaces can be deployed for any defined domain. Part of the requirements to support such a generalised application is to automate the deployment of the interaction against the given ontologies as much as possible. In order to support this automation, we have been investigating the formalisation of the mSpace interaction model which informs CS AKTive Space [18].

It is important to note that the mSpace model is visualisation agnostic. The model supports particular affordances such as rearranging the organisation of the information structure, that can be embodied in a variety of visualisations. CAS represents a partial instantiation of an mSpace with a particular visualisation - in this case a multi-column view that supports *dimensional sorting* or the ability to move a column to a different location in the view, effectively reorganising the hierarchical ordering of the information presented. In describing the properties of an mSpace deployed in CAS, we used the naming conventions of the multi-column representation described above in order to overview the attributes of the model below.

As described above and in [1], mSpace and AKTive Space interfaces consist of a series of user interface controls, which we call *columns*, each of which presents a selection of objects for the user to choose from. A user's interaction with these columns progresses from the left to the right. The selection that they make in a column affects the selections which are offered in the subsequent columns that lie to its right. The selection offered by each column is a collection of objects of the same conceptual type; a list of people, or of institutions, or of publications, for example. Each of the columns represents some facet of the objects being searched, and a selection within a column can be thought of as specifying some dimension in the sparse multidimensional space of a knowledge base (where each class is treated as a dimension).

In this respect, the interface resembles the menu-based interfaces used for exploring databases; a user chooses from lists of options that lead the user through the formulation of a query. The mSpace method differs in that a context (or *column layout*) may be dynamically reconfigured so as to place one dimension at a higher importance than others.

Each column in an interface contains a class of objects which share some common characteristics. We represent this class by a class expression in a description logic-based language such as OWL [19]; the objects presented in a column are the extension of the class expression for that column. The interactions between columns take the form of constraints which determine the choice being presented in each column and the effect that a selection has on other columns. These constraints are also expressed as class expressions in the description logic language. The overall class expression for a column is the intersection of the separate constraint class expressions that apply to that column.

A selection in a column affects the choices that are presented in other columns. Selection constraints represent the relation between a particular column and the other columns in a layout (or between one class of objects and another class of objects in the ontology), and are applied to a column in addition to any other constraints it may have.

Selection constraints take the form of a restriction on a property which relates the two columns, to the value which was selected. These restrictions are typically (but not always) `owl:hasValue` restrictions, written in description logic terms as $\exists R.\{s\}$, where s is the selected value and R is the relating property. Multiple selections in a column are handled by forming the union of the constraints for the individual selections (or by adding both selections to the enumerated class, as in $\exists R.\{s_1, s_2\}$). Selection constraints are particular to a given pair of column types; a constraint which works between publications and authors will not work between publications and publishers, because the underlying relations which link these types are different.

The set of constraints for a layout must provide a means to relate each column to every other column, either directly, or indirectly by chaining more than one constraint together. For a layout with n columns, the *basis set* of constraints contains $n-1$ constraints. It is possible to construct systems with more than this minimum number of basic constraints, but this raises the possibility of inconsistency when there is more than one way to relate one column to another, possibly by chaining constraints, and those relations are incompatible or have different meanings.

The characterisation of an mSpace user interface in terms of description logic expressions should be considered to be an abstract description of such a user interface, and not a detailed specification of an implementation itself.

In CS AKTive Space, we have chosen to limit the class expressions used as selection constraints to be `owl:hasValue` restrictions. We have further chosen to implement these constraints in a naïve manner which does not require recourse to a description logic reasoner; constraints are represented as triple patterns containing variables which are bound as selections are made. For example, a constraint $\exists$ *works-at*.*Institution* (which relates a column containing researchers to one containing institutions) is replaced with the triple patterns $(\textit{?person}, \textit{works-at}, \textit{?institution})$ and $(\textit{?institution}, \textit{rdf:type}, \textit{Institution})$. These triple patterns are expressed in RDQL, and used as queries to the triplestore which generate the values with which the columns are populated.

This approach has the advantage of simplicity, because a constraint and its inverse have exactly the same form. The triple patterns do not assume that the constraint will be read in the direction of a relation or of its reverse; the reading of the constraint depends on which variable is bound (by a selection), and which is free (and generates potential selection with which a column is populated).

The disadvantage of this implementation approach is that it permits only one particular type of constraint, and that the construction and composition of disjunctive query patterns (as would be the case when multiple selections are made in a column), is clumsy in RDQL.

IV. Further Work

To build this large-scale Semantic Web application we needed to use simple solutions to a number of the scientific and technical issues; we have therefore identified a significant number of areas for future research, which we are now pursuing.

We are in the process of writing DAML Services [20] descriptions for the services shown in Figure 1, so that future revisions of the application will be able to opportunistically discover and make use of new services within the system, and adapt to the withdrawal of existing services.

CS AKTive Space is targeted at a particular domain, that of CS research in the UK. We are now working on generalisations to AKTive Space, so that the system can work in a domain-independent fashion. CS AKTive Space is based on a single ontology, although that ontology has been built up from a number of different sources. In the future, the AKTive Spaces will work with multiple heterogeneous ontologies, either by coalescing them or mapping between them, or a combination of both.

CS AKTive Space has also only implemented dimensional sorting in the interface. The mSpace model supports other operations that enable further interaction customisation. These include the ability to replace one dimension with another (swap out Researcher for Project, for instance) or to add new dimensions (add Project or Awards) or remove dimensions (look at only Region and Researcher). Future iterations of AKTive Space applications will include more of these model features.

Further challenges for the Semantic Web have also been identified. What are the best ways to sustain an acquisition and harvesting activity? How best should the harvested content be modelled? How should we cope with the fact that there are bound to be large numbers of duplicate items that need to be recognised as referring to the same objects or referents? The ability of the inferential services to scale as more content becomes available will prove an issues sooner or later. What are the good ways to present the content so that inherent patterns and trends can be directly discerned? How trustworthy is the provenance and accuracy of the content? How is all this information to be maintained and sustained as a social and community exercise?

Finally, the way in which further services might enhance the user interaction is still open; the use of Armadillo and Ontocopi illustrates how services of particular types can work with AKTive Spaces, but the use of other services can be expected to require other interactions with AKTive Space.

To conclude, we believe that our construction of the CS AKTive Space application has demonstrated a significant aspect of the Semantic Web: namely that it is possible to meet many of the challenges described above and deliver the ability to explore an ontologically-mediated information space gathered from a wide set of heterogeneous sources.

References

[1] m.c. schraefel, M. Karam, and S. Zhao, "mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia," in *Proceedings of the AH2003 Workshop on Adaptive Hypermedia and Adaptive Web-Based Systems*, Nottingham, UK, June 2003, pp. 217–235.

[2] H. Alani, S. Dasmahapatra, K. O'Hara, and N. Shadbolt, "Ontocopi: Using ontology-based network analysis to identify communities of practice," *IEEE Intelligent Systems*, vol. 18(2), pp. 18–25, 2003.

[3] S. Kampa, "Who are the experts? e-scholars in the semantic web," Ph.D. dissertation, University of Southampton, 2002.

[4] T. Leonard and H. Glaser, "Large scale acquisition and maintenance from the web without source access," in *Workshop 4, Knowledge Markup and Semantic Annotation, K-CAP 2001*, Oct 2001, pp. 97–101.

[5] F. Ciravegna, A. Dingli, D. Guthrie, and Y. Wilks, "Integrating information to bootstrap information extraction from web sites," in *Proceedings of IJCAI03 Workshop on Information Integration, held in conjunction with the International Conference on Artificial Intelligence (IJCAI-03)*, August 2003.

[6] AKT. (2002) The AKT Reference Ontology. [Online]. Available: http://www.aktors.org/publications/ontology/

[7] S. Harris and N. Gibbins, "3store: Efficient bulk RDF storage," in *Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03)*, 2003, pp. 1–20. [Online]. Available: http://eprints.aktors.org/archive/00000273/

[8] "Advanced Knowledge Technologies project," 2000. [Online]. Available: http://www.aktors.org/

[9] Hewlett-Packard Labs, "RDQL - RDF data query language," Hewlett-Packard Labs, Tech. Rep., 2003. [Online]. Available: http://www.hpl.hp.com/semweb/rdql.htm

[10] A. Dingli, F. Ciravegna, and Y. Wilks, "Automatic semantic annotation using unsupervised information extraction and integration," in *Proceedings of SemAnnot 2003 Workshop*, 2003.

[11] H. Alani, S. Dasmahapatra, N. Gibbins, H. Glaser, S. Harris, Y. Kalfoglou, K. O'Hara, and N. Shadbolt, "Managing reference: Ensuring referential integrity of ontologies for the semantic web," in *Proceedings 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02)*, 2002, pp. 317–334.

[12] I. Niles and A. Pease, "Towards a standard upper ontology," in *Proceedings of the Second International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.

[13] D. Schaffer, Z. Zuo, S. Greenberg, L. Bartram, J. Dill, S. Dubs, and M. Roseman, "Navigating hierarchically clustered networks through fisheye and full-zoom methods," *ACM Transactions on Computer Human Interaction (TOCHI)*, pp. 162–188, 1996.

[14] C. Plaisant, B. Shneiderman, K. Doan, and T. Bruns, "Interface and data architecture for query preview in networked information systems," *ACM Transactions on Information Systems (TOIS)*, vol. 17, no. 3, pp. 320–341, 1999.

[15] R. Chimera and B. Shneiderman, "An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents," *ACM Transactions on Information Systems (TOIS)*, vol. 12, 1994.

[16] B. Bloom, *A Taxonomy of Educational Objectives, Handbook 1: Cognitive Domain*. McKay, 1965.

[17] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins, "Polyarchy visualization: visualizing multiple intersecting hierarchies," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press, 2002, pp. 423–430.

[18] N. Gibbins, S. Harris, and m. schraefel, "Applying mspace interfaces to the semantic web," 2003, preprint. [Online]. Available: http://triplestore.aktors.org/tmp/www2004-mspace-model.pdf

[19] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," World Wide Web Consortium," Proposed Recommendation, Dec. 2003. [Online]. Available: http://www.w3.org/TR/owl-features/

[20] The DAML Services Coalition, "DAML-S: Semantic Markup for Web Services," May 2003. [Online]. Available: http://www.daml.org/services/0.9/