CS AKTive Space: Building a Semantic Web Application

Hugh Glaser[1], Harith Alani[1], Sam Chapman[2], Alexiei Dingli[2], Nicholas Gibbins[1], Stephen Harris[1], m.c. schraefel[1], and Nigel Shadbolt[1]

School of Electronics and Computer Science, University of Southampton, Southampton, United Kingdom
nrs@ecs.soton.ac.uk

Department of Computer Science, University of Sheffield, Sheffield, United Kingdom

Abstract. In this paper we reflect on the lessons learned from deploying the award winning [1] Semantic Web application, CS AKTiveSpace. We look at issues in service orientation and modularisation, harvesting, and interaction design for supporting this 10million-triple-based application. We consider next steps for the application, based on these lessons, and propose a strategy for expanding and improving the services afforded by the application.

1 Introduction

In 2000, the Advanced Knowledge Technologies (AKT) project was launched as a collaborative project between a number of UK universities (Southampton, Aberdeen, Edinburgh, Sheffield and the Open University). The funding was £7.5m and the project was planned to last for six years. The aim was to bring together a range of different disciplines to investigate the issues that were then emerging where large-scale networks and data sources were meeting more intelligent processing than hitherto. Many of these issues have now been captured by the term "Semantic Web" which has gained currency since Tim Berners-Lee's article in Scientific American [2].

As the AKT project progressed, the partners from the different disciplines captured and advanced the state of the art in each area, and a number of Technology Integration Experiments (TIEs) were planned to ensure that the different technologies informed each other. The full range of individual technologies can be found at [3]. One of these TIEs has led to the subject of this paper, the CS AKTiveSpace.

CS AKTiveSpace exploits the wide range of semantically heterogeneous and distributed content that has been collected relating to Computer Science research in the UK. This content is gathered on a continuous basis using a variety of methods including harvesting and scraping as well as adopting a range of models for content acquisition. The content currently comprises around ten million

RDF triples. We made use of the storage, retrieval and maintenance methods we developed in the first two years of the project to support its management. The content is mediated through an ontology constructed for the application domain that was defined in the project, and incorporating components from other published ontologies. CS AKTiveSpace interaction supports the exploration of patterns and implications inherent in the content and exploits a variety of visualisations and multi dimensional representations. Knowledge services supported in the application include investigating communities of practice: who is working, researching or publishing with whom (through ONTOCOPI), and on-demand acquisition: requesting that further knowledge be found from the web about a person. This phase of the project culminated in CS AKTiveSpace winning first place in the Semantic Web Challenge, 2003.

In the following sections, we consider in particular, harvesting, service orientation and modularisation, and user interaction. We look in particular at two of the core services that have been integrated: ONTOCOPI and Armadillo. ONTOCOPI is a community of practice service while Armadillo is a natural language harvesting service used to supplement our triplestore with further information about entities of interest at the user's request. We conclude with a discussion of the lessons learned in these components of the project and look at how these will be focusing our efforts for the next three years.

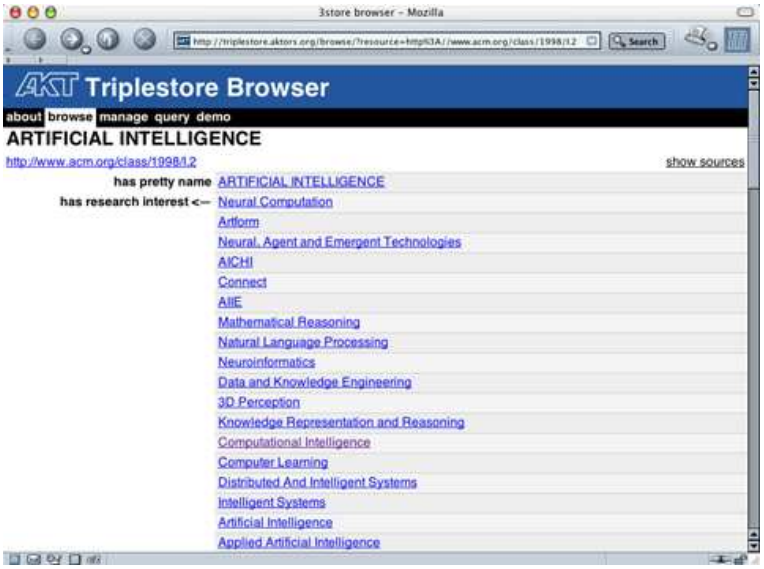

Fig. 1. Browser

2 Ontology Development

The information sources that are aggregated and presented by the CS AKTiveSpace are mediated through a common ontology. This ontology, the AKT Reference Ontology, was developed within the AKT project over a six month period, drawing heavily on ontologies written earlier by AKT partners. The reasons for this were 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 mutually inconsistencies, which raises the cost of deciding upon appropriate mappings between ontology 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 AKTiveSpace 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 AKTiveSpace. The selection of elements to be displayed in CS AKTiveSpace 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.

The chief concern with ontology choice is that of the resulting performance of the AKTiveSpace 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.

3 Service Orientation and Modularisation

3.1 Harvesting

To be convincing, the building of a Semantic Web application requires data on a large scale. 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 heterogeneous sources for our application domain were not available. We therefore devoted effort from the outset 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.

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. 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 AKTiveSpace 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 use a relatively scruffy approach in which 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 AKTiveSpace. 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 (the viral, rich-get-richer phenomenon that we later describe needs to start somewhere). 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 hyphen metadata gathered by these mediators consists of 430MB of RDF/XML files containing around 10 million RDF triples describing 800,000 instances of people, places, publications and other items of interest to the academic community.

3.2 Armadillo

Information can be present in different formats on the Web, in documents, in repositories, (e.g. databases or digital libraries). From such resources or their output, it is possible to extract information with different reliability. Systems such as databases generally contain reliable structured data and can be queried using an API or an Induced Wrapper. Wrapper Induction methodologies are particularly suited to model rigidly structured Web pages such as those produced by databases [5], [6]. When the information is contained in textual documents, extracting information requires more sophisticated methodologies. Wrapper induction systems have been extended to cope with less rigidly structured pages [7], free texts and mixtures of them [8]. The degree of complexity increases from rigidly structured to free texts. The more complex the task is then generally the less reliable the extracted information is. Also, as complexity increases more training data is required. Wrappers can be trained with a handful of examples whereas full IE systems may require millions of words [9]. The more the task becomes complex, the more information is needed for training, the more reliable input data becomes difficult to identify.

Armadillo [10] is a service for on-the-fly, user-determined, directed knowledge acquisition, which is used to opportunistically expand a given knowledge base. Armadillo makes use of a simple methodology, which populates ontologies from

	Possbile	Actual	Correct	Wrong	Missing	Precision	Recall	F-Measure
Seed	320	152	151	1	168	99.3	47.2	64.0
IE-based	320	217	214	3	103	98.6	66.9	79.7

Table 1. Armadillo paper discovery

distributed information. The system starts from a set of generic strategies, typically wrappers, defining where to search for initial information and what to look for. When data is harvested using these strategies, it is passed to an oracle, such as a reliable database, which is used to verify the results to an initial degree of certainty. Armadillo works because of the inherent redundancy of information on the web. Redundancy is given by the presence of multiple citations of the same information in different contexts and in different superficial formats. This factor is currently used for improving question answering systems [11]. When recognized information is present in different sources, it is possible to use its multiple occurrences to bootstrap recognisers that, when generalized, will retrieve other pieces of information, producing more generic recognisers [12]. The system loops in this manner to discover more and more reliable information.

The Armadillo AKTive Space Demo application works in the following way. First a user submits an armadillo request to populate the underlying ontology via the AKTive Space site. This request is automatically submitted to a remote Armadillo server, which queues requests until one of many Armadillo clients is available to process it. An Armadillo client uses triplestore RDFQL queries to determine what knowledge is already asserted regarding the instance to be populated, and to help resolve referential integrity issues [13]. To prevent compounding of any errors this does not collate information that has been previously submitted from Armadillo. With a basic ontology of information, each entity is rated for reliability by utilising the redundancy of its sources. If information has been elicited from numerous sources it is assumed to be more reliable than an entity from only one source. With an aim to locate web pages that contain information suitable for incorporating into the ontology, higher rated entities are used to search for likely pages of interest, using the Google search engine. Armadillo uses Amilcare for Information Extraction to identify the structure of found pages so that additional entities can be identified. This approach finds additional content for the ontology and each entry is again rated for reliability, this process continuously cycles until a set timeout has occurred. Finally the populated ontology is asserted into the AKT Triplestore in the form of RDF triples. In this way, future requests for information about the instance in question will potentially return more information.

The Armadillo approach of initially wrapping followed by repeated Information Extraction can be seen to dramatically improve the recall of available data with a minimal loss of precision. This is demonstrated in table 1 showing the performance when eliciting papers for eight individuals, this is detailed further in [10].

3.3 ONTOCOPI

ONTOCOPI (ONTOlogy-based Community Of Practice Identifier [14]) is a tool that finds sets of similar instances to a selected instance in a knowledge-base. If an ontology (i.e. both the classification structure and the knowledge base of instantiations) represents the objects and relations in a domain, then connections between the objects can be analysed. The aim of ONTOCOPI is to extract patterns of relations that might help define a Community Of Practice (COP). COPs are informal groups of individuals interested in a particular job, procedure or work domain, who swap insights on tasks connected with work [15]. COPs are very important in organisations; taking on important knowledge management (KM) functions. Increasingly, COPs are being harnessed by organisations to carry out aspects of their KM (for an extended example, see [16]). Community identification is currently very resource-heavy, often based on extensive interviewing. ONTOCOPI supports the task of COP identification by analysing the patterns of connectivity between instances in the ontology. The ONTOCOPI hypothesis is that the informal relations that define a community of practice can be inferred from the presence of more formal relations (such as being the member of a group, the author of a paper, and so on). For instance, if A and B have no formal relation, but they have both authored papers (formal relation) with C, then that indicates that they might share interests (informal relation). It is our working hypothesis that the COPs identified by ONTOCOPI can be decent proxies for communities of practice.

Algorithm The ONTOCOPI expansion algorithm generates the COP of the selected instance by identifying the set of close instances and ranking them according to the weights they accumulate from several path traversals. It applies a breadth first, spreading activation search, traversing the semantic relations between instances (ignoring directionality) until a link threshold is reached. The link threshold allows COPs to be of different ranges. A narrow COP of a project may consist of only those entities in direct relation to the project (e.g. project employees, member organisations, themes), while a wider COP may include indirectly related entities, such as the colleagues of the project's members, other projects about the same theme or subject, etc. Consider the example in figure 2. Assume we need to identify the COP of the query instance A, using the relationships *has Author*, *memberOf* and *attended*, with the weights 1.0, 0.6, and 0.3 respectively. All instances will have an initial weight of 1. Activation will spread from the query instance to neighbouring instances in the network, up to a given number of links. In the first expansion, the query instance A will pass on weight to all the instances it is connected to. The amount of weight passed equals the weight of the instance multiplied by the weight of the traversed relationship. In this case, A passes 1×0.6 to D, and 1×1 to H. These will be added to their initial weights of 1. In return, these instances will pass their total weights to all their neighbours, so D for example will pass $(1 + 1 \times 0.6) \times 0.6$ to B and A. Expansion will stop when the link paths are exhausted or the link threshold is

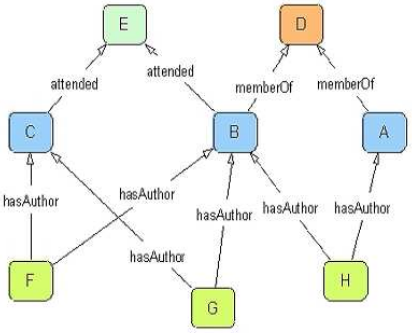

Fig. 2. Example Ontology Network

reached Instances therefore accumulate weight based on the number of relevant relations they have with the initial instance.

3.4 ONTOCOPI in CS AKTiveSpace

The complete implementation of ONTOCOPI exists as a plugin for Protégé 2000 [17] as presented in Alani et al[14] and O'Hara et al [18]. A less sophisticated version (with less features) has been developed as a web service and linked to the 3Store. ONTOCOPI can access the ontology and knowledge base stored in the 3Store via RDQL queries. When a person instance is selected within CS AKTive-Space, it sends the URI of this instance in a HTTP query to ONTOCOPI, which in turn queries the 3Store to identify the CoP of the given instance. The result will be a list of person URIs that forms the community of the selected individual. The list is returned back to CS AKTiveSpace as an RDF file.

As explained earlier, community identification in ONTOCOPI is based on analysing knowledge networks of instances and their interconnections. Hence its quality is directly dependent on that of the knowledge source. When integrating ONTOCOPI with CSAS, it was noticed that the quality of the resulting COPs was mostly affected by three main factors in the 3Store knowledge base:

1. Richness: This is related to the amount and variety of knowledge in the 3Store. For example if only a small amount of information is available about a person, and of limited variety (e.g. few authored papers), then the identified COP for this individual will inevitably be shallow and of poor quality. The richness of the 3Store is always on the increase as more knowledge is being harvested from an increased number of sources.
2. Connectivity: Performance of ONTOCOPI will be compromised if the network to be analysed is fragmented and heavily disconnected (i.e. a large number of connections are missing between existing instances). Acquiring more knowledge to establishing further connections is therefore important for COP identification.

3. Duplication: One of the persisting problems often encountered when gathering knowledge from multiple sources is that of duplication. Information about the same individuals or objects is often acquired from various sources. It is necessary to have a mechanism for identifying and resolving duplications to maintain the integrity of the knowledge base and the quality if the services it provides. ONTOCOPI's output greatly improved when *sameAs* attributes where added to the 3Store. ONTOCOPI's algorithm was modified to take into account these relations when crawling the knowledge network.

4 Server Implementation

4.1 Scale

Due to the volume of data (approximately 15M RDF triples) and degree of interlinking and inference required the RDF data and RDFS schema required storage in a dedicated RDF(S) server application, rather than queries run directly over RDF/XML files. It would have been possible to implement the back-end directly over RDF/XML data with no caching or pre-parsing, but the computational cost would be high enough to prevent interactive use.

The scale of the data combined with the use of RDFS' inferential capabilities required some advances in the area of RDFS storage, these are further described in [19], but mainly involve an efficient RDBMS to RDF mapping and hybrid eager/lazy entailment generation.

The scalability of the system was tested further by asserting another 10M triple knowledge base, and the system showed no obvious difference in performance, but the ability to scale to level of data describing international research will likely require more work in the area of efficiency and scalability.

4.2 Expressivity

For the most part RDFS was found to be adequately expressive for the description of the resources used in the CSUK data, however there is no way to indicate the equivalence of two RDF resources, so we required the OWL sameAs property for these uses. This property is particularly common where we have multiple documents describing the same individuals, such as the RAE's description of a particular researcher combined with their home institutions description.

The correct handling of this co-reference situation is just one small part of the problem it poses, a larger, and in many ways still unsolved part is the correct automatic identification of co-referent entities.

There is a more subtle issue relating to excessive over analysis – and general excessively expressive structures – in ontological structures. For example the Calendar-Date class in the AKT Reference Ontology[20] is structured as shown in figure 3.

For cases where the interface wishes to display a date in a form understandable by a use this requires a moderately complex query to be executed, whereas a simple string, or XML Datatype date representation can be retrieved and parsed with a single, simple query.

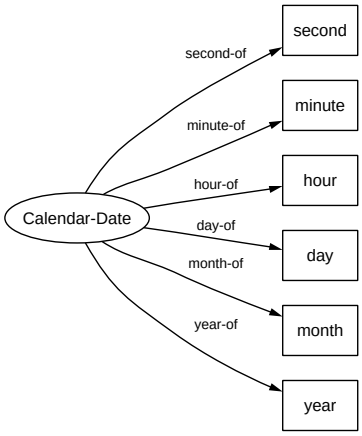

Fig. 3. AKT Reference Ontology date representation

5 User Interface

The CAS interface lets users explore the domain of Computer Science in the UK. Users can look at particular regions of the country or research areas as defined by the ACM classification system to see who in a given region is working in which areas. Users can also explore a given researcher's community of practice, as well as get a sense of where that person ranks in terms of funding level in an area or other UK rating criteria. The CAS interaction design emphasises exploration of a domain via a persistent context of the associated attributes in the domain

The CS AKTiveSpace interface is informed by the mSpace model for representing n-dimensional domains for interaction [21]. In this model, users are presented with a default projection through the space. When the projection is flattened, the result is that the projection is flattened into a hierarchy. These flattened projections are referred to as "slices." An mSpace, allows certain manipulations on the hierarchy: users can rearrange the ordering of elements in the slice, changing both the dependencies in the hierarchy. This allows users to determine the focus of interest for a given slice. Similarly, the mSpace model supports swapping dimensions in a current slice, adding dimensions and subtracting dimensions. With these manipulations, users can extend or contract a slice, as well as effectively change the projection itself. There are constraints within the model to test whether a swap, addition or subtraction is sensible in the context of the current slice.

mSpaces also emphasise the maintenance of context. Instances in a given dimension of a slice are persistent: selection of an instance reveals the next level of the hierarchy without occluding the previous level of the hierarchy. In this way both the path through the hierarchy and the context of the path elements are persistently available. Similarly, the selection of an instance at any point in the path provides information about that currently selected instance. This

Fig. 4. CS AKTiveSpace

associated information is made available to improve the context of selection so that the user has more information than simply the label of an instance in order to assist decisions about navigation through the space.

The current CAS has implemented only a subset of the mSpace functions. It presents a default slice, and users can reorganise the dimensions in the slice to support their focus of interest. Users cannot at this time swap, add or subtract dimensions. They do however get detailed information about any instance selected in a given dimension. So, selecting a given researcher, for instance, provides a brief profile of the researcher: contact information, top publications and community of practice information.

One of the reasons why the CAS interaction is an imperfect mSpace is that, while we knew what the affordances of the model were, we had not yet formalised how best to implement these in a semantic web application. Indeed, we used our efforts to instantiate some of the core mSpace functions as a way to better refine the model and its relation to representing it within OWL. The pragmatic consequences of this design learning experience are that we are faced with the question of whether or not to redeploy CAS to utilise the proper model and gain the additional affordances.

In terms of good interaction design, there are other attributes that we also need to incorporate into the UI, both to support better access and persistent provenance.

- State and History. We do not currently capture state, so that, for instance, if users have an information state in the interface which they wish to share with someone else, they have no way to do this without reiterating the steps for getting to that state. State is fairly straightforward to implement. History is not. History would allow users to step through their exploration of a domain. This replay would allow them to compare sessions, paths and provenance of an exploration
- Order: We do not order the attributes in a dimension in any specific way. Adding ordering on each dimension (currently represented by a column) would give the added query on that dimension to see, for instance, either a listing by alphabet or by ranking of the institutions in the list, according to Research Assesment Exercise, or by grant total and so on.
- Number: a simple value to provide per dimension/column is how many elements are in a current dimension/column at any point.

The above list represents refinements to the extant UI. That UI represents an interaction paradigm that is distinct from the Web's primary keyword search/ link click search/browse interaction. We have been focusing on the benefit of this exploration approach, which we view as a complement to search/browse, not as a replacement. As such, an integrated interface would provide both exploration and search/browse. It is not obvious how keyword search would be integrated with the CAS UI so that users would be able to see the both the context of a given retrieved item as well as the list of possible hits.

The CAS UI has also shown us that, even with 10 million triples, we do not have sufficient content to provide a truly rich reflection of the domain modelled. The expressive power of these applications, in order to make the kinds of comparisons between X and Y comes with scale. Harvesting data can only take us so far, it seems. So, while we have demonstrated with CAS a new kind of representation and interaction for Semantic Web applications, we are seeking ways to improve content acquisition so we can better utilise the model's capabilities.

Overall, the interaction design for CAS has been well received. Demonstrations of CAS have motivated various industrial, government and funding groups to solicit us on building AKTive Spaces for aspects of their intranets. That the interaction design is well-formalised we anticipate will help to make this translation process both tractable and effective.

6 Client Implementation

The CS AKTiveSpace client is implemented as an XHTML and ECMA-Script web page, rather than using a piece of external software, such as a Java Applet. We felt that this technique was more in-keeping with the spirit of the semantic web, as it builds on existing web technologies.

6.1 Data access and query execution

Queries are run against the RDF(S) used to inform the client using RDQL [22] queries in HTTP GET requests. The HTTP services queried return XML documents containing the RDQL binding table that satisfies the query.

The scripts running in the web page initialise the display by querying the schema and instances that are required to build the initial state of the interface. For the CSUK application these initial queries can be time consuming over slower Internet links, sometimes taking up to 30 seconds, as they require the transfer of a significant amount of data spread over several hundred requests. If the interface was being rolled out as a production service, with relatively a static schema it would have been advantageous to cache the data required to initialise the client - or pass it in some bulk, compressed form to reduce the initialisation time.

6.2 mSpace column implementation

The design on the column interactions is based upon mSpace [23], but the implementation in RDF required a mapping between the structures present in the RDF schema and the columns.

The mapping chosen for the implementation of this version is the most direct one mapping directly from column to column in the order they appear on the screen. Although this mapping is quite easy to understand it produces different results for the same query if expressed in a different order – for example. if a research area is chosen before a region, you will not necessarily get the same results as when the region is chosen before the research area.

This implementation has led to a formalisation and more detailed explanation of the column/semantic mapping [21].

7 Concluding Remarks

This paper has discussed the technologies and structure that forms the CS AKTiveSpace. We began by discussing the ontology design choices, and noted that what is required is a sufficiently expressive ontology, and that the structure influences the performance of the application, rather than the capabilities. We then discussed the harvesting of data for the application. For the CS AKTiveSpace to be realistic, it was important that a wide range of heterogeneous sources were used; until the semantic web becomes a reality, it will be necessary to build and use targeted tools to acquire data from appropriate sources for particular application areas.

We then discussed one of the services utilised by CS AKTiveSpace - Armadillo. This natural language processor enables the user of the semantic web application to request and initiate further knwoledge discovery from the web, as required. Another service was then presented - ONTOCOPI. This service gives the user a sophisticated tool for finding out complex relationships between instances in the space.

The amount of data for an application such as this is high, and we included a section on the server, noting that it can now serve 25M RDF triples in an efficient manner.

In the last section, the most obvious aspect of the application was discussed, the User Interface. This was introduced through the theory behind it, followed by comments on the implementation.

Finally, CS AKTiveSpace shows that a large-scale ontology mediated application can recruit, in a dynamic fashion. heterogenious information resources and resent them in a tractable and efficient manner. The principled composition of these semantic services is the subject of ongoing research within the AKT project.

8 Acknowledgements

This work was supported by the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC). The AKT IRC is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01 and comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

References

1. Semantic Web Challenge 2003. http://challenge.semanticweb.org/ (2003)
2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
3. The AKT project: The AKT Technologies. http://www.aktors.org/technologies/ (2003)
4. Leonard, T., Glaser, H.: Large scale acquisition and maintenance from the web without source access. In: Workshop 4, Knowledge Markup and Semantic Annotation, K-CAP 2001. (2001) 97–101
5. Kushmerick, N., Weld, D.S., Doorenbos, R.B.: Wrapper induction for information extraction. In: Intl. Joint Conference on Artificial Intelligence (IJCAI 1997). (1997)
6. Muslea, I., Minton, S., Knoblock, C.: Wrapper induction for semistructured web-based information sources. In: Proceedings of the Conference on Automatic Learning and Discovery (CONALD 1998). (1998)
7. Freitag, D., Kushmerick, N.: Boosted wrapper induction. In: Proceedings of the 17th National Conference on Artificial Intelligence (AAAI 2000). (2000)
8. Ciravegna, F.: Adaptive information extraction from text by rule induction and generalisation. In: Proceeedings of the 17th International Joint Conference On Artificial Intelligence (IJCAI 2001). (2001)
9. Yan, T., Garcia-Molina, H.: The sift information dissemination system. In: ACM TODS 2000. (2000)
10. Dingli, A., Ciravegna, F., Wilks, Y.: Automatic semantic annotation using unsupervised information extraction and integration. In: Proceedings of SemAnnot 2003 Workshop. (2003)

11. Dumais, S., Banko, M., Brill, E., Lin, J., Ng, A.: Web question answering: Is more always better? In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002). (2002)
12. Brin, S.: Extracting patterns and relations from the world wide web. In: WebDB Workshop at 6th International Conference on Extending Database Technology, (EDBT 1998). (1998)
13. Alani, H., Dasmahapatra, S., Gibbins, N., Glaser, H., Harris, S., Kalfoglou, Y., O'Hara, K., Shadbolt, N.: 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) 317–334
14. Alani, H., Dasmahapatra, S., O'Hara, K., Shadbolt, N.: Ontocopi - using ontology-based network analysis to identify communities of practice. IEEE Intelligent Systems **18(2)** (2003) 18–25
15. Wenger, E., McDermott, R., Snyder, W. Harvard Business School Press, Cambridge, Mass (2002)
16. Smith, R., Farquhar, A.: The road ahead for knowledge management: An ai perspective. American Association for Artificial Intelligence **21(4)** (2000) 17–40
17. Musen, M., Fergerson, R., Grosso, W., Noy, N., Grubezy, M., Gennari, J.: Component-based support for building knowledge-acquisition systems. In: Proceedings of the Intelligent Information Processing (IIP 2000) Conference of the International Federation for Processing (IFIP), World Computer Congress (WCC'2000). (2000) 18–22
18. O'Hara, K., Alani, H., Shadbolt, N.: Identifying communities of practice: Analysing ontologies as networks to support community recognition. In: Proceedings of the IFIP World Computer Congress (IFIP 02), Montreal, Canada. (2002)
19. Harris, S., Gibbins, N.: 3store: Efficient bulk RDF storage. In: Proceedings of the 1st International Workshop on Practical and Scalable Semantic Systems (PSSS'03). (2003) 1–20 http://eprints.aktors.org/archive/00000273/.
20. The AKT project: The AKT Reference Ontology. http://www.aktors.org/publications/ontology/ (2002)
21. Gibbins, N., Harris, S., schraefel, m.: Applying mspace interfaces to the semantic web. preprint: http://triplestore.aktors.org/tmp/www2004-mspace-model.pdf (2003)
22. Hewlett-Packard Labs: RDQL - RDF data query language. http://www.hpl.hp.com/semweb/rdql.htm (2003)
23. schraefel, m., Karam, M., Zhao, S.: mSpace: Interaction design for user-determined, adaptable domain exploration in hypermedia. (2003) 217–235