SKUA – retrofitting semantics

Norman Gray[1 2], Tony Linde[1] and Kona Andrews[3]

[1] Department of Physics and Astronomy, University of Leicester, UK,
[2] Department of Physics and Astronomy, University of Glasgow, UK
[3] Institute for Astronomy, University of Edinburgh, UK

Abstract. The Semantic Web promises much for software developers, but because its claimed benefits are rather abstract, there is little obvious incentive to master its unfamiliar technology. In contrast, many 'Social Web' applications seem rather trivial, and not obviously useful for astronomy.
The SKUA project (Semantic Knowledge Underpinning Astronomy) is implementing a service which will realise the benefits of both these web technologies. This RESTful web service gives application authors ready access to simple persistence, simple (social) sharing, and lightweight semantics, at a low software-engineering cost. The SKUA service allows applications to persist assertions (such as bookmarks and ratings), and share them between users. On top of this, it provides lightweight, astronomy-specific, semantics to enhance the usefulness and retrieval of the users' data.

1 Introduction

For all its current fashionability, we can identify at least two reasons why the Semantic Web excites little interest among astronomical software developers. Firstly, there is so far no well-known 'killer app' for the semantic web, and the use-cases sometimes brandished in support of the Semantic Web's promise – involving machines booking hospital appointments, or comparing prices ([1], and see `http://www.w3.org/2001/sw/`) – are not obviously relevant to astronomical applications development. Secondly, even when a potential application is dimly discernable – and everyone can agree it must *somehow* be useful for a machine to 'know' that a black hole is a type of compact object – there are multiple barriers of novel terminology, standards and technology to be overcome before an idea can be turned into a useful software product. This can be a significant technology hurdle for an application developer who may be rationally sceptical about the practical utility of semantic web technologies.

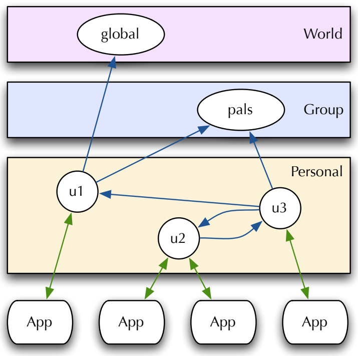
In the SKUA project (`http://myskua.org`) we are developing an infrastructure which addresses both of these concerns. The SKUA infrastructure provides a mechanism for persisting and sharing a flexible range of application state, including annotations (of which we give examples below), in a way which lets applications transparently take advantage of lightweight semantic knowledge within the SKUA system. That is, we are helping application developers painlessly 'retrofit' lightweight semantics to their existing applications at those points where they already persist some data, or could do. By combining the social aspects of the annotation sharing and the lightweight semantics, the SKUA infrastructure can be regarded as a simple 'Web 3.0' application, to the extent that that term represents the anticipated melding of Web 2.0 applications with Semantic Web technologies.

2 The SKUA infrastructure

The SKUA infrastructure consists of a network of assertion services, each node of which is an RDF triple store and SPARQL endpoint [2], currently implemented using Jena; these are referred to as SACs, or 'Semantic Annotation Collections', and can be either on separate servers or logically distinct entities on a single server. These are the objects to which applications write per-user state information – 'assertions' – such as annotations ('this paper is good'), or preferences ('I'm interested in pulsars'). The annotations can then be retrieved using a SPARQL query by the same application, by another instance of the same application, or by a cooperating application.

The infrastructure also allows these assertions to be shared between users, in such a way that an application's SPARQL query against its 'local' service is forwarded to the services it federates to (see Fig. 1). This is naïve federation, in which the SAC simply forwards the query to its peers, which potentially forward it in turn, with the results merged before being returned to the caller; thus the final result is the union of the query results to the various nodes, rather than the result of the query over the union of the nodes. Though limited, we believe this model is reasonable in this case, since the information in the various nodes is likely to be both simple and relatively homogeneous. Thus if, in Fig. 1, user 'u1' shares the assertion that 'paper X is good', then when an application belonging to user 'u3' looks for good papers, it picks up the corresponding assertion by 'u1'. This query federation will be permitted only if the user making the assertion explicitly allows it (which is important in the case where the assertion is something like 'the author of paper Y is clearly mad').

Our permissions model is very simple. Each SAC is a personal utility, conceptually more like a USB stick with a PIN than a service with a username/password pair. Each SAC is configured with a list of the SACs to which it should forward queries, and a list of the SACs from which it should accept forwarded requests. Here, the SACs identify themselves when making a delegated query, and do not do so with any delegated credentials from the user on whose behalf they are making the query. This model is easy to implement, we believe it is easy for humans to reason with, and since SACs and users have a close relationship, a user's SAC is an adequate proxy for the user themself. This federation model supports both a tree-like and a peer-to-peer network, or anything in between, while allowing a client application to ignore this structure and query only the user's personal SAC. The trust model is simple-minded, and keeping private my opinion about 'the author of paper Y' depends on my friends not federating



```
@base <http://blah/my-sac>.
@prefix s:
  <http://myskua.org/claimtypes/1.0/>.
@prefix dc:
  <http://purl.org/dc/elements/1.1/>.
<#b1>
  a s:bookmark;
  s:ref [
    a s:webpage;
    s:url <http://www.w3.org/2001/sw/>;
    dc:title "SemWeb @ W3C";
    s:extended "The W3C nexus"
  ];
  s:tag "semanticweb", "rdf";
  s:time "2008-01-21T18:11:58Z".
```

Fig. 1. SKUA's sharing architecture: on the left we show the relationships, both peer-to-peer and hierarchical, between annotatation stores, with double-headed arrows indicating read-write relationships with applications, and the single-headed arrows indicating the federation of queries between services; and on the right we illustrate a potential annotation type, in this case a URL bookmark, using the Turtle notation for RDF [3].

carelessly. User interface details will help couple the user's mental model to the actual model, but only experience can tell us if the trust model is fundamentally too simple in fact.

Since federation consists only of passing on a SPARQL query, a SAC can federate to any SPARQL endpoint. We have not yet discovered how useful this will be in practice.

Although we have observed that the nodes have astronomy-specific knowledge built in, this is only due to astronomy-specific TBox information uploaded at configuration time, and though this project is specifically motivated by astronomy, the architecture is nonetheless general.

2.1 Interfaces

The SKUA SACs are updated and queried via a RESTful API.

The various annotations are modelled as fragments of RDF which are each named by a URL; these are referred to as 'claims' within a SAC. Although the project has defined a lightweight ontology for claims, the RDF which composes a claim is unrestricted. Claims are created by POSTing the RDF to the SAC URL, which responds with a freshly-minted URL naming the claim, which can of course be retrieved in the obvious fashion, with a GET. The contents of the SAC can also be queried by POSTing a SPARQL query to the SAC URL. Individual claims can be replaced by PUTting fresh RDF to the claim URL. There is no cross-reference between the various claims – at least in the applications we have envisaged so far – so little scope for linked-data cross-linking.

The network of federations, and the type of reasoning available (using any of the reasoners available in Jena, or none), is controlled by SAC metadata, also in the form of RDF. This is set when the SAC is created, and may be later adjusted using the Talis Changeset Protocol (`http://n2.talis.com/wiki/Changeset_Protocol`).

The API is described in a WADL specification available at (`http://myskua.org/doc/qsac/`). Independently of any (debatable) use of this specification for generating client code, we find it useful for generating the interface documentation and generating support for regression tests.

2.2 Implementation

The SAC is implemented using Jena (`http://jena.sourceforge.net`) and SISC (`http://sisc-scheme.org/`), and runs as a web service either standalone (using Jetty), or within a Tomcat container. Essentially all of the application logic is written in Scheme, which allows for rapid development and which, being almost entirely functional, is well-suited for web applications.

The SKUA software is available at `http://skua.googlecode.com`. The current version, at the time of writing, supports updating, persistence, querying and federation; vocabulary-aware querying is available but undocumented; easier sharing and security are in development; and the design of a more sophisticated authorisation model awaits deployment experience.

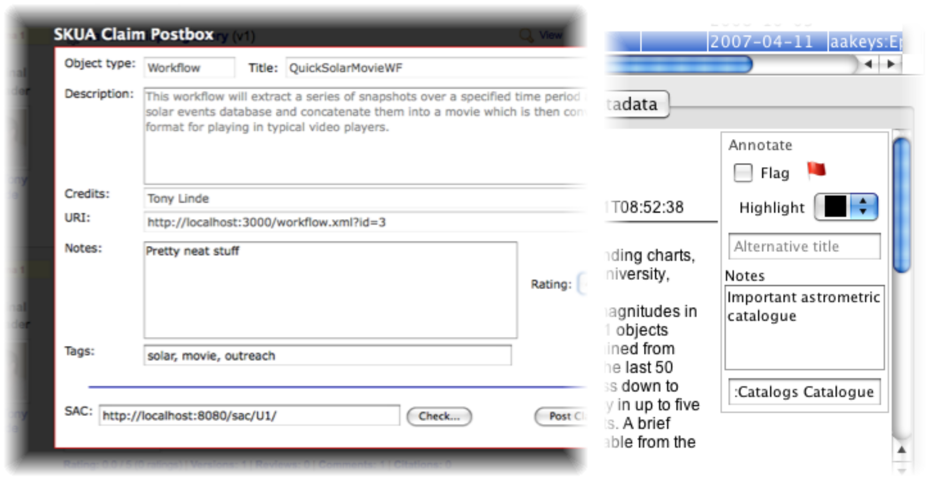
3 Example applications

An important aim of the SKUA project is to develop applications which use the project's infrastructure, both as a way of validating the approach, and for their intrinsic usefulness. As well, we are cooperating with the developers of existing applications to support them in adding SKUA interfaces where appropriate.

In particular, we are developing *Spacebook* [4], as an adaptation of the myExperiment code-base ([5], see also `http://myexperiment.org/`). This allows scientists to share digital objects of various kinds, supporting the development of communities. Spacebook builds on this by adding integration with AstroGrid's Taverna workflows, and lets users tag resources using the SKUA infrastructure.

As well, we have adapted the AstroGrid registry browser, VOExplorer [6]. The International Virtual Observatory Alliance (IVOA, `http://www.ivoa.net`) is a consortium of virtual observatory projects, defining and deploying consistent interfaces for accessing astronomical data services. These service resources – image archives and catalogues – are registered in an IVOA registry, and VOExplorer is one of a small number of user-facing applications which allow astronomers to browse the registry, and search within it, including the free-text keyword fields included in the curation metadata.

For each Registry entry, VOExplorer displays title, description, curation and other information, and provides a simple interface for the user to specify a highlight colour, notes about the resource, an alternative title, and tags (see Fig. 2).


SKUA Claim Postbox

Object type: Workflow Title: QuickSolarMovieWF

Description: This workflow will extract a series of snapshots over a specified time period
solar events database and concatenate them into a movie which is then con
format for playing in typical video players.

Credits: Tony Linde

URI: http://localhost:3000/workflow.xml?id=3

Notes: Pretty neat stuff

Rating:

Tags: solar, movie, outreach

SAC: http://localhost:8080/sac/U1/

2007-04-11 aakeys:E

T08:52:38

ding charts,
niversity,
agnitudes in
objects
ned from
e last 50
s down to
in up to five
s. A brief
ble from the

Annotate

Flag

Highlight

Alternative title

Notes

Important astrometric catalogue

:Catalogs Catalogue

Fig. 2. Annotation panels for Spacebook (left) and VOExplorer (right)

In its original, default, mode, the application persists this information to a local file, but it can also be configured to persist the information to a SKUA SAC; this is not yet the default because SACs have not yet been deployed sufficiently broadly to make this useful to most users.

Users can tag resources using any tags they please, but if they attach keywords from one of the existing IVOA vocabularies [7] a subsequent search on the SKUA store is able to take advantage of the lightweight semantics associated with these keywords. For example, if a user annotates a resource with `aakeys:Ephemerides`, they can later make a SPARQL query for terms which have `AstrometryAndCelestialMechanics` as a broader term, and in doing so pick up resources tagged with `Astrometry`, `CelestialMechanics`, `Eclipses`, `Ephemerides`, `Occultations`, `ReferenceSystems` or `Time`.

The Paperscope application (`http://paperscope.sourceforge.net/`) is a utility for searching and browsing ADS (`http://adswww.harvard.edu/`), which is the principal bibliographic database for astronomy and astrophysics. Like VOExplorer, Paperscope has a simple tagging interface, and like VOExplorer, it was originally limited to a single machine. We have started work on extending the application to use the SKUA RDF nodes as a simple persistence service, using the existing UI and interaction model.

Both the VOExplorer and Paperscope applications were provided with tagging support rather as an afterthought, and in both cases this was barely developed because the tagging could not be shared. Replacing the simple file-handling code with the barely-more-complicated SKUA interface, without changing the user interfaces at all, means that the applications can immediately share annotations and take advantage of the lightweight vocabulary reasoning which the SAC provides. It is in this sense that we claim that the semantic technologies have been *retrofitted* to the applications, giving them an immediate injection of semantic functionality with minor investment in implementation code, and so allowing the authors to experiment with the user-oriented functionality which this semantic technology prompts.

We emphasise that we are not expecting users to write SPARQL queries for themselves, but instead expect applications to issue them on the user's behalf, based on simple query templates. To support this extra functionality, application developers need make no major commitments to semantic web technologies, and need only manage HTTP transactions using (readily templatable) RDF such as that in Fig 1, and basic SPARQL queries.

4 Conclusion

We have described a simple architecture for storing and sharing simple RDF annotations of external resources, using a RESTful interface to a SPARQL endpoint. The interface is such that application developers have a low barrier to entry, and need make few technology commitments before reaping the benefit of simple semantic enhancement of their applications. We are deploying support for the architecture in a number of existing applications.

Acknowledgements

The SKUA project is funded by the UK's Joint Information Systems Committee (`http://www.jisc.ac.uk`).

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 2001)
2. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Candidate Recommendation (June 2007)
3. Beckett, D.: Turtle - terse RDF triple language. W3C Team Submission (January 2008)
4. Linde, T., Gray, N., Andrews, K.: Spacebook: resource sharing for astronomers using SKUA technology. In Bohlender, D., Dowler, P., Durand, D., eds.: Astronomical Data Analysis Software & Systems, XVIII, PASP (2009)
5. De Roure, D., Goble, C.: myExperiment – a web 2.0 virtual research environment. In: International Workshop on Virtual Research Environments and Collaborative Work Environments, Edinburgh. (2007)
6. Tedds, J.A., Winstanley, N., Lawrence, A., Walton, N., Auden, E., Dalla, S.: VO-Explorer: Visualising data discovery in the virtual observatory. In Argyle, R.W., Bunclark, P.S., Lewis, J.R., eds.: Astronomical Data Analysis Software and Systems, XVII. Volume 394. (2007) 159
7. Gray, A.J.G., Gray, N., Hessman, F.V., Martinez, A.P.: Vocabularies in the virtual observatory. IVOA Proposed Recommendation (2008)