

SKUA Final Report

Norman Gray and Tony Linde, University of Leicester
Kona Andrews, University of Edinburgh

r504 of 2009-10-18T16:50:41

Contents

1 Introduction

- 1.1 Background p2
- 1.2 Deliverables p3
- 1.3 JISC pro-forma final report p3

2 SKUA architecture

3 Implementations

- 3.1 Spacebook p5
- 3.2 SAC server p5
- 3.3 Client software p6

4 Dissemination

- 4.1 Documentation and standards development p7

5 Project management and project experiences

- 5.1 Institutions and personnel p7
- 5.2 Evaluation p7
- 5.3 Implementation experiences p8

A Standards

B Technical development – agile methodology

- B.1 The process p9
 - B.1.1 Background p9
 - B.1.2 Practices p10

References

Executive Summary

The SKUA Project (Semantic Knowledge Underpinning Astronomy) has implemented a distributed architecture of semantically aware RDF stores. This ‘semantic layer’ supports a cluster of applications which either directly support users in finding and recovering useful resources, or indirectly support them by supporting user-facing applications. Although the system is somewhat specialised to astronomy, and proved by its interaction with, and eventual embedding within, the Virtual Observatory, the bulk of the semantic knowledge is localised in the RDF store, with the design goal that it could be replaced if desired by the analogous semantic knowledge of a different domain.

Specifically, the project:

- developed the SKUA infrastructure using Semantic Web technologies, and delivered a server-side application usable both as a multi-user central service or user-installable desktop servers;
- validated the approach and APIs by developing new tools or adapting existing ones to add semantic annotation sharing; and
- in particular, produced a ‘Spacebook’ application, to support social networking between astronomers.

The best compact summary of the project, containing the relevant technical details and motivation, is reference [7].

1 Introduction

1.1 Background

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 ([2], 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.

Astronomy has been part of the UK’s e-Science effort since its inception, the majority of this under the AstroGrid project¹. The focus of this effort, in the UK and within projects in at least 15 other countries, is the creation of a world-wide Virtual Observatory (VO), making astronomical data and applications easily available to astronomers regardless of their location and affiliation. The VO will, by defining and implementing standard interfaces, make it possible to access common resources from multiple applications. These resources are located via a globally distributed resource registry, which has been defined and working for over two years now.

The Virtual Observatory (VO) is a world-wide collaboration, supporting astronomical research through a network of projects to support data management, interoperability, portable workflows and common services. It is managed at the international level by the International Virtual Observatory Alliance (IVOA), acting as a standards body closely modelled on the W3C. The UK has a long-term leading role in the VO through the UK e-Science AstroGrid project, AstroGrid participation in the European VO Project, and the substantial UK investment in the European Southern Observatory (ESO), another Euro-VO partner. A primary focus of the various international VO projects is the continuing definition and maintenance of practical and internationally supported metadata describing archive data and web services; and one focus of the SKUA project is to add semantic value to the deployed VO metadata registries, aligned with ongoing VO efforts to develop ways of making these registries useful to astronomical applications.

¹<http://www.astrogrid.org>

The PIs are strongly connected with the VO's development plans, and are in a position to react quickly to, and support, the needs of VO application authors.

The VO has an existing distributed registry service, containing metadata about large numbers of resources, from organisations and institutions, to large-scale data archive services. This registry is deployed already, in the form of a network of database-backed services.

The global VO has long recognised both the necessity and the complexity of shared metadata, and has made substantial time and software investments in the VO registry network described above. It recognises, however, that the problem is not yet completely solved, and is moving towards semantic solutions compatible with the solutions in this JISC proposal. This proposal therefore represents an opportunity to give a JISC-funded project a leadership role in the design of a component crucial to the infrastructure of the UK, European and world-wide VO efforts.

1.2 Deliverables

The SKUA bid included the following deliverables

- Deliverable D2.1** Set up RDF version of VO registry. This was produced as a proof of concept, but we have no plans to support the service long-term.
- D2.2** Implement SAC, building on well-established triplestore implementation. See Sect. 3.2.
- D2.3** Develop thin user-facing client for simple SAC management and setup. See Sect. 3.3.
- D2.4** Produce initial API documentation for client authors. See Sect. 3.3.
- D3.1** Develop spacebook application, and refine interface through regular releases. See Sect. 3.1.
- D4.1 & D4.2** Develop 'suggestions service', and plugins. During the course of the project, it became clear that, although a suggestions service would be a reasonable potential user of a SKUA SAC, this functionality exists at a higher level than was our immediate priority.
- D5.1** Papers for appropriate semantic web and astronomy conferences. See Sect. 4.
- D5.2** Contributions to astronomy-specific publications (in particular IVOA Standards). See Sect. 4.1.
- D5.3** SKUA final report, including discussion of applications outside astronomy. This is delivered by this present document.
- D5.4** Organise a workshop on the project outcomes. Although it was not a SKUA workshop as such, the Semantic Astronomy 2009 workshop was organised by one of the project PIs, and featured two presentations covering the SKUA work (see Sect. 4).

1.3 JISC pro-forma final report

The mapping to the JISC template final report is as follows

Acknowledgements The SKUA project was funded by the E-Infrastructure programme of JISC, and implemented in the Department of Physics and Astronomy, University of Leicester, and at the Royal Observatory Edinburgh.

Executive Summary See above.

Background See Sect. 1.1.

Aims and Objectives See the list of deliverables in Sect. 1.2.

Methodology See Sect. B.

Implementation Discussed in Sect. 5.3.

Discussion of XP in Sect. B.1.

Outputs and Results The project's outputs consist of the software and web pages described in Sect. 3, and the publication outputs described in Sect. 4.

Outcomes See Sects. 5.2 and 5.3.

Conclusions See Sect. 5.3 for further discussion.

Implications We believe that this is a generalisable architecture, and can be further exploited at low cost. We anticipate, and are currently working on developing, further projects which use the SKUA project architecture and software.

Recommendations (optional) See Sect. 5.3.

References See the references section at the end.

2 SKUA architecture

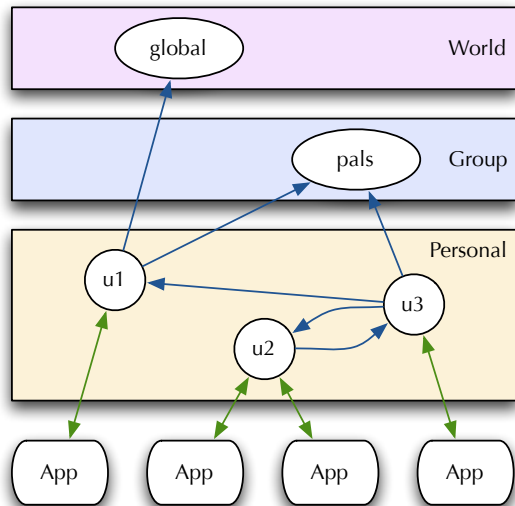


Figure 1: The SKUA architecture

The core component is a network of Semantic Assertion Collections (SAC) providing rather generic semantic Web Services. For performance reasons, we expect the semantic reasoning within the SACs to be rather simple, with more elaborate reasoning either performed in the background and separately asserted, or simply retained within value-adding clients. The optimal level of integration with, or even replacement of, the VO registries, will become clear during the course of the project.

This structure integrates with e-Infrastructure outcomes by supporting **new ways of retrieving data**, and by **integrating with key initiatives in the wider research community**.

We conceive the semantic layer as a directed acyclic graph (DAG) of SACs, each of which can store a greater or smaller number of RDF triples and, crucially, federate queries to a configurable list of partner stores, in such a way that a query against one SAC is effectively made

against the RDF triples stored in that SAC and all the SACs that it federates to (Fig. 1). Thus the personal SAC, which may be a local desktop service or a personal section of a remote service, will typically store user-specific annotations or notes, and the global SAC will store VO-wide information such as an RDF mirror of the VO Registry. Information is transparently shared by being copied from a local SAC to an appropriate one of the SACs shared within a research group, or an ad-hoc group of collaborators, with this copy process being managed, directly by the user, using a small UI, or as a part of a separate user-facing application's functionality.

Each SAC has a (standard) SPARQL endpoint which will respond to queries both from clients and from other SACs which federate to this one. Each SAC will also support a simple RESTful API for managing its RDF data.

A SAC must not respond to queries indiscriminately, since to do so would expose possibly private annotations; each SAC will keep a list of those SACs to which it has permitted federation. The topology of federations is specified exclusively by the SACs which do the federation; the permission to query or to write to a SAC is the responsibility of the SAC being federated to. The VO is deploying a SSO/Security infrastructure which this project would make use of. This infrastructure would handle the authentication issues involved, but we anticipate leaving the SAC access-control as the responsibility of the SACs themselves (either internally, or at the HTTP layer if appropriate).

We believe these three functions – querying, updating and sharing RDF information – will support a flexible and open-ended array of user-supporting client applications, and we will validate this assertion by developing an initial set of such applications, as described below.

The SKUA project uses standard standard technologies and protocols, composed in an innovative way. The SACs build on one of multiple available triple-store implementations, and they are queried using the W3C-standardised SPARQL query language (<http://www.w3.org/TR/rdf-sparql-query/>). The VO security infrastructure realises JISC investments by building on the Shibboleth infrastructure. The simple SAC management interface will be specific to the SACs, but there will be no requirement for this to go beyond the standard REST interaction pattern. Our goal is to produce a simple, open-source, and easily composable, Web Service, proved by applications. This builds on the PIs' experience with generations of application/service deployments in the VO and other projects.

3 Implementations

The following sections provide brief summaries of the project's implementation work. For fuller discussion, see [7].

All of the code described below is maintained in the project's publicly available Subversion repository, at <http://skua.googlecode.com>.

3.1 Spacebook

An important aim of the SKUA project was 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 developed *Spacebook* [10], 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.

3.2 SAC server

One of the core SKUA software outputs was the server which holds the annotations – the Semantic Annotation Collection, or SAC. This was developed at the start of the project, based on earlier work for the Astrogrid project². This service provides its functionality through a RESTful interface³.

²The earlier work is now available at <http://quaestor.googlecode.com>

³The interface is documented at <http://myskua.org/doc/qsac/interface-http.html>



Figure 2: Annotation panels for Spacebook (left) and VOExplorer (right)

3.3 Client software

As discussed above, in Sect. 3.2, the SAC server has a RESTful interface. This is an important architectural choice, because it is emphatically independent of choices of platform and architecture. While it is not hard to use, it does require an application to implement a certain amount of relatively low-level plumbing. To lessen the inconvenience here, we also implemented a thin Java client library which provided a Java-friendly object interface on top of the RESTful one.

As well as the project-specific application Spacebook (see Sect. 3.1, we validated the spacebook design by adapting two existing, and deployed applications for the astronomical community.

As well, we have adapted the AstroGrid registry browser, VOExplorer [12]. 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). 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 [6] 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 tag-

ging 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 Dissemination

The project's dissemination activities took place online and in relevant academic conferences.

The project website at <http://myskua.org> provided an overall introduction to the project, and user-facing documentation. At the same time, the project source code and wiki was hosted at <http://skua.googlecode.com>.

Publications [8, 10] were delivered to an astronomy conference, and [7] to a semantic web conference. At the time of writing, the project also has an abstract submitted to the UK e-Science All-Hands meeting.

Although it was not a SKUA workshop as such, the 2nd Practical Semantic Astronomy workshop⁴ was organised by one of the SKUA PIs, part-sponsored by the SKUA project, and featured two presentations describing the SKUA work (see <https://dspace.gla.ac.uk/handle/1905/806> for downloads).

4.1 Documentation and standards development

There were no standards developed as part of the SKUA work, though there is a link to the ongoing IVOA work on developing SKOS-based vocabularies [6]. The SKUA project used and validated these vocabularies during its testing.

5 Project management and project experiences

5.1 Institutions and personnel

The project institutions and personnel were:

University of Leicester, Dept. Physics and Astronomy Norman Gray (project manager), Tony Linde

Royal Observatory Edinburgh Kona Andrews

5.2 Evaluation

The SKUA project was a success. The project produced all the principal deliverables listed in Sect. 1.2. The only significant deliverable not completed was the

⁴<http://www.practicalastroinformatics.org/conferences/semast09>

‘suggestions service’, and this was because, as noted in Sect. 1.2, this was felt to be at the wrong level for this project to reasonable implement.

The Spacebook application (Sect. 3.1) has not had a great deal of take-up. This can be attributed to the very great challenges involved in promoting a social application, which we felt was beyond both our remit and our expertise.

We have added SKUA-client functionality to two existing applications (Sect. 3.3). In both cases, the adaptations have been accepted into those applications’ main code bases, albeit disabled by default. This is because there is a chicken-and-egg problem here: without deployed SKUA SACs, there is little motivation for users to enable this functionality, but without this functionality enabled, there is little motivation to deploy the SACs. This becomes a social-networks problem, as above, but we hope to address it in future projects using the SKUA software and services.

The main goal of the Spacebook application and the adapted existing applications was to demonstrate that the SKUA SAC was indeed usable for the purposes we claimed, and that developers could use our documentation to code against it. We feel that we have comfortably shown this to be true.

5.3 Implementation experiences

The project proceeded very smoothly.

The agile methodology discussed below in Sect. B didn’t work as magically as we had hoped, but it achieved the core goal of helping the distributed team (in three locations, and meeting face-to-face only two or three times during the project) act in concert. Overall, it was clear that this was a supportable model, which achieved its goals in outline, but which needed a little more social tweaking before it became as effective as the approach promises.

We have described a simple architecture for storing and sharing simple RDF annotations of external resources, using a RESTful interface to a SPARQL endpoint. Crucially, 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.

Recommendations:

- The SKUA project has addressed the take-up problem described at the beginning of Sect. 1.1; the shape of the solution has been illuminated by the SKUA project’s results, but the problem is not fully solved, and if JISC wishes to explore, and encourage take-up of, Semantic Web technologies in the near future, this take-up gap should be further explored.
- JISC should support projects in investigating agile methodologies for project development.

A Standards

The SKUA project made use of the following standards or best practices.

RDF : a network of W3C Recommendations for supporting the integration and exchange of knowledge on the Web <http://www.w3.org/RDF/>.

SPARQL : a W3C Recommendation for making structured queries to RDF triple-stores <http://www.w3.org/TR/rdf-sparql-query/>.

Agile methodologies : a set of best practices for managing projects (most typically software design projects) in which the design and, to some extent, goals are not fixed at the start of the project, but are instead developed and delivered in relatively short iterations through the life of the project (see Sect. B and Sect. B.1).

Linking Open Data : a set of best practices for linking semantically enhanced data, via RDF, on the Web [3].

B Technical development – agile methodology

The project will use an agile development methodology. In such a methodology, the project outcomes and design are specified beforehand only in broad detail, and the detailed planning and development is instead framed in terms of shorter-period *iterations* of, in our case, three months.

In an agile methodology, the project aims to produce a product with at least some basic functionality as quickly as possible, and make it available to users immediately (in our case, the users are VO application authors). Then, at the beginning of each iteration, the project identifies which functionality it could add next, based on specific user-stories. It then selects those features which can be reasonably added in a single iteration, implements those, and produces another release at the end of the iteration. The advantages of this methodology are as follows.

- There is a functioning product at almost all times.
- Therefore the product can be confronted with users early and often, while there is still time to amend the design and interface. User feedback and deployment experience influence the design of the project at all stages, and faulty architectural decisions can be identified early and fixed in time.
- When planning each iteration, the project can use the difference between the expected and actual implementation times of the previous iteration, and thus improve its time estimates for the next iteration.

The SKUA project used a variant of the eXtreme Programming (XP) methodology adapted to a distributed development process – with thanks for guidance from Neil Chue Hong and Ross Gardler (of OMII-UK and OSS-Watch respectively). This methodology is described in more detail in Sect. B.1 below.

B.1 The process

The following describes and, we hope, justifies the agile process we followed in the SKUA project. Of necessity, we adjusted the process to suit our situation, rather experimentally.

B.1.1 Background

SKUA development used an agile development methodology, instead of the traditional heavyweight process: specifying requirements, architecture, design, implementation, testing, and finally discovering that what works isn't what's wanted, and what was wanted doesn't work.

Agile methodologies in general aim for a very tight coupling between requirements, implementation and release, cycling round the three in iterations as short as a week. They are characterised by having very little up-front design, and having a functional and releasable system at all times, which is incrementally expanded in the light of direct 'customer' requirements.

Specifically, we used a suitably-adjusted variant of the Extreme Programming (XP) methodology, building on various sources [1, 9, 11].

The reason for using a variant is twofold. Firstly, XP methods are typically described as working for teams of between four and 20 programmers, while we

had two or three times 50% FTE; and because a notable feature of XP is its emphasis on co-location, to the extent of suggesting that all actual code is written by pairs of programmers sharing a single keyboard.

One cannot simply delete XP practices at random, however. The set of XP practices form a principled and coherent whole, and if we remove one practice, we must aim to replace it with one which has the same purpose. An important group of the XP practices which presume co-location are actually concerned primarily with communication, both between the members of the team and communicating the status of the project in a very immediate way.

B.1.2 Practices

The following are XP practices which are both clearly useful to us, and adaptable to a distributed team. We expect that both the list, and the way we adapt them to our situation, will change over the course of the project.

Visions To counteract the centrifugal force of XP's focus on short-term goals, such projects benefit from having an explicit statement of the project's overall goals. So yes, we have visions.

'Customers' XP relies on the notion of a 'customer', who takes an active role in the development process. The 'customer' is a representative of the person or entity who is going to receive the value, or benefit, of the delivered product. Their role is to act as a walking, talking, requirements document, ready to generate or elaborate stories, decide on priorities, and inform the programmers about how they wish to use the final product. Our intended 'customers' were two applications written by AstroGrid colleagues (Paper-scope and VODesktop), and we consulted with those applications' authors where necessary; however these authors couldn't be expected to devote a lot of time to our project, and so the project development team ended up acting as their proxies when discussing user stories.

User stories One of the core XP notions is the notion of user stories, which are brief accounts of discrete blocks of functionality, described from the point of view of a user of the system, and small enough that they can be implemented in one or two days work. We maintained a list of UserStories, from which we selected a set to implement at the regular iteration meeting.

The planning game is a precursor to the iteration planning meeting. At base, programmers and customers work through the list of extant stories, with the customers prioritising unimplemented stories, and programmers estimating the effort required to implement.

Iterations The other core XP notion is the iteration. At the beginning of an iteration, the group selects a set of user stories which they plan to implement in that iteration. At the end, they review what has been completed, comparing actual to expected progress, and release the improved software. That is, the software is re-releasable at the end of every iteration, and is released in fact on release dates decided well in advance.

XP suggests iterations of one or two weeks. Since the SKUA programmers were generally working only 50%, we felt that two-week iterations would be best. We used Skype to have iteration meetings.

The iteration planning meeting consists of:

- Retrospective of previous iteration (demo, number of stories completed, etc)

- Select stories to include in next iteration. Based on the programmers' estimates of required effort, the meeting selects a number of stories whose effort sums to the effort represented by the set of stories completed in the last iteration.

Informative workplace As part of its group of communication practices, XP promotes the notion of the informative workplace. Here, the status of the project – in terms of the stories pending and committed to for an iteration, planned release dates, the group's velocity (the number of stories usually completed per iteration), and perhaps outstanding bugs – is made as immediately visible as possible. As an initial attempt at this, we used a google-doc document which each of the project members could write to.

Stand-up meetings A common XP practice – part of the group focused on communication – is the stand-up meeting. This is a daily meeting in which each participant reports very briefly – taking around 30 seconds – on what they got done yesterday, what they plan to do today, and what problems are blocking them. Like the informative workplace practice, this is intended to let status information percolate through the group, making it possible for all members to contribute to solving problems.

Although physical proximity is, again, one of the key components of this practice in the XP methodology, got at least some of the relevant benefit from regularly scheduled brief Skype conversations.

References

- [1] Kent Beck and Cynthia Andres. *Extreme Programming Explained*. Addison-Wesley, 2004.
- [2] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, May 2001.
- [3] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data -- the story so far. *International Journal On Semantic Web and Information Systems*, 2009. To appear (special issue on Linked Data). Available from: <http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf>.
- [4] David Bohlender, Patrick Dowler, and Daniel Durand, editors. *PASP*, 2009.
- [5] Dave De Roure and Carole Goble. myExperiment -- a web 2.0 virtual research environment. In *International Workshop on Virtual Research Environments and Collaborative Work Environments*, Edinburgh, 2007.
- [6] Alasdair J G Gray, Norman Gray, Frederic V Hessman, and Andrea Preite Martinez. Vocabularies in the virtual observatory. IVOA Proposed Recommendation, 2008. Available from: <http://www.ivoa.net/Documents/latest/Vocabularies.html>.
- [7] Norman Gray, Tony Linde, and Kona Andrews. SKUA - retrofitting semantics. In Sören Auer, Chris Bizer, and Gunnar Aastrand Grimnes, editors, *Proc. 5th Workshop on Scripting and Development for the Semantic Web (SFSW) at ESWC 2009, Heraklion, Greece.*, volume 449 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2009. Available from: <http://ceur-ws.org/Vol-449/ShortPaper5.pdf>.
- [8] Norman Gray, Tony Linde, and Kona Andrews. The SKUA project and the semantic web. In D.A. Bohlender, D. Durand, and P. Dowler, editors, *Astronomical Data Analysis Software and Systems (ADASS XVIII, Quebec, Canada)*, volume 411, pages 175--178. ASP Conference Series, 2009.

- [9] Neil Chue Hong and Ross Gardler. Open and agile development [online]. 2008. Available from:
<http://www.slideshare.net/rgardler/agile-and-open-development>.
- [10] Tony Linde, Norman Gray, and Kona Andrews. Spacebook: resource sharing for astronomers using SKUA technology. In Bohlender et al. [4].
- [11] James Shore and Shane Warden. *The Art of Agile Development*. O'Reilly Media, Inc., 2007.
- [12] J. A. Tedds, N. Winstanley, A. Lawrence, N. Walton, E. Auden, and S. Dalla. VOExplorer: Visualising data discovery in the virtual observatory. In Robert W Argyle, Peter S Bunclark, and James R Lewis, editors, *Astronomical Data Analysis Software and Systems, XVII*, volume 394, page 159, 2007. Available from:
<http://adsabs.harvard.edu/abs/2008ASPC..394..159T>.



<http://myskua.org>

Copyright 2009, University of Leicester. This work is licensed under the Creative Commons Attribution-Share Alike 2.0 UK: England & Wales Licence. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-sa/2.0/uk/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.