SWAD-Europe deliverable 12.1.7: Semantic Portals Demonstrator- Lessons Learnt

Project name:
Semantic Web Advanced Development for Europe (SWAD-Europe)
Project Number:
IST-2001-34732
Workpackage name:
12.1 Open Demonstrators
Workpackage description:
http://www.w3.org/2001/sw/Europe/plan/workpackages/live/esw-wp-12.1.html
Deliverable title:
Semantic portals demonstrator - lessons learnt (demo_2_report)
URI:
http://www.w3.org/2001/sw/Europe/reports/demo_2_report/
Authors:
Dave Reynolds, HP Laboratories, Bristol, UK
Paul Shabajee, Graduate School of Education and ILRT, Bristol, UK
Steve Cayzer, HP Laboratories, Bristol, UK
Damian Steer, (Contractor) HP Laboratories, Bristol, UK
Abstract:
The Semantic Portals demonstrator is the second of the two open demonstrators which make up workpackage 12.1. In this report we summarize the ideas behind the demonstrator and describe the design, implementation and deployment of the demonstrator. At each stage we try to identify lessons that have been learnt from the demonstrator.
The demonstrator itself may be visited at: http://www.swed.org.uk.
Status:
Initial release.
Comments on this document are welcome and should be sent to Dave Reynolds or to the public-esw@w3.org list. An archive of this list is available at http://lists.w3.org/Archives/Public/public-esw/

Contents

1 Introduction

This report is part of SWAD-Europe Work package 12.1: Open demonstrators. This workpackage covers the selection and development of two demonstration applications designed to both illustrate the nature of the semantic web and to explore issues involved in developing substantial semantic web applications.

This report concerns the second demonstrator *Semantic Portals*. In this demonstrator we take the notion of a decentralized information portal, built using semantic web tools and standards, and apply it to a test problem domain. Our chosen test domain is a directory of environmental and biodiversity organizations. This specific demonstration is referred to as SWED, for *Semantic Web Environmental Directory*, throughout the report.

The details and rationale for this choice of demonstrator are described in detail in our earlier report [REQUIREMENTS] though we begin this report with a short summary of these details for completeness.

The bulk of this report describes the architecture, design and implementation of the demonstrator and lessons learnt from the process. We cover both the ontology/thesaurus designs needed to represent our chosen domain and the software components needed to implement the live demonstrator. These components comprise a web *portal* that supports visualization and navigation of the RDF descriptions, an *aggregator* that fetches updated RDF descriptions from organization web sites and a *data entry tool* for creating new RDF descriptions.

We then discuss the reaction of example organizations to the notion of the semantic portal and to the SWED demonstrator. In general that reaction has been extremely positive. The specific SWED demonstrator has gained good feedback from key coordinating organisations such as the UK Environment Council - so much so that discussions are underway on how the service could be maintained and expanded after the close of the SWAD-Europe project. The demonstrator has also been very effective as an illustration of the Semantic Web. In particular, it convinced the Natural History Museum that the exchange and aggregation of data using semantic web standards was both a useful and practical approach and as a result they have begun to explore its applicability to collection-level descriptions.

2 The semantic portals demonstrator

The goal of the open demonstrator work package is to provide two demonstration semantic web applications which illustrate the nature of the semantic web. In particular, in our initial analysis and selection report [ANALYSIS] we identified the three key features of the semantic web to illustrate as being data representation, explicit semantics and webness. The Semantic Portals demonstrator was specifically chosen to illustrate a balanced mix of all three of these defining characteristics.

2.1 Features of the semantic portal approach

We use the term Semantic Portal to refer to an information portal in which the information is acquired and published in semantic web format and in which the structure and domain model is made explicit (e.g. in the form of published ontologies).

There are several advantages to using semantic web standards for information portal design. These are summarized in Table 1 and expanded below.

Traditional design approach	Semantic Portal
Search by free text and stable classification hierarchy.	Multidimensional search by means of rich domain ontology.
Information organized by structured records, encourages top-down design and centralized maintenance.	Information semi-structured and extensible, allows for bottom-up evolution and decentralized updates.
Community can add information and annotations within the defined portal structure.	Communities can add new classification and organizational schemas and extend the information structure.
Portal content is stored and managed centrally.	Portal content is stored and managed by a decentralized web of supplying organizations and individuals. Multiple aggregations and views of the same data is possible.
Providers supply data to each portal separately through portal-specific forms. Each copy has to be maintained separately.	Providers publish data in reusable form that can be incorporated into multiple portals but updates remain under their control.
Portal aimed purely at human access. Separate mechanisms are needed when content is to be shared with a partner organization.	Information structure is directly machine accessible to facilitate cross-portal integration.

Figure 1 - contrast semantic portals proposal with typical current approaches

Ontologies -

The use of an explicit, shared domain ontology enables both data sharing and richer site structure and navigation including multidimensional classification and browsing schemes. Use of the Semantic Web standards for encoding these ontologies also enables the ontologies themselves to be shared and reused across portals. Several projects have already derived benefits from ontology-driven portal designs [⊣ SEAL][⊣ WEB-PORTALS].

Evolution -

Requirements change over time leading to extensions to the information model. The semantic web helps in two ways. Firstly, the user interface and submission tools can be generated from the declarative ontology. Secondly, the semi-structured data representation of RDF permits new data properties and types to be incrementally added without invalidating existing data, in such a way that both original and extended formats can be used interchangeably. This suggests an alternative approach to information portal design. Instead a long top-down design cycle, we start from a seed ontology and information structure that we extend incrementally.

Community extensions -

Whilst many portals support constrained community annotations, such as comments and ratings, the semantic web approach allows more extensive community customization. For example, during work on a portal for wildlife multimedia it became clear that many user communities would like specialized navigation of the data (based on formal species taxonomy or behavior depicted), which was unfeasible for the centralized portal provider. Using the decentralized approach it is possible for communities to develop these specialist navigation structures as a set of external RDF annotations on the portal data. The central site can then aggregate the community-provided enrichments.

Aggregation and decentralization -

One problem with traditional information portals is that they are often dependent on the responsiveness of the central maintainers, so that if funding disappears, so may the data. In the semantic web approach supplying groups host their own data and the portal becomes an aggregating service. Central organization is still needed (for example, to provide the initial impetus and ensure that appropriate ontologies and controlled vocabularies are adopted). However, once the system reaches a critical mass it can more easily be self-sustaining - anyone can run an aggregator service and ensure continued access to the data or a new supplier can add data to the pool without a central organization being a bottleneck.

3.2 The Semantic Web Environmental Directory (SWED) demonstrator

To illustrate these advantages in practice and to build a functioning demonstrator we needed to pick a domain for the demonstrator portal. As the demonstrator domain we chose to develop a directory of UK environmental, wildlife and biodiversity organisations. We termed this specific demonstration service the Semantic Web Environmental Directory, abbreviated to SWED throughout the rest of this report.

The idea is that each organization wishing to appear in the directory provides their organization description as RDF data, using a web-based data entry tool, and then hosts the data at their own web site (similar in style to FOAF [⊣ FOAF]). A portal aggregates the RDF data and provides a faceted browse interface to allow users to search and browse the aggregated data. Annotations to this data can be created by third parties and hosted by the suppliers or by an annotation server. These annotations permit new classification schemes and relational links to be added to the data. In particular, the ability to add new links is seen as opening up exciting opportunities to capture and visualize the complex relationships between environmental organizations.

For more background on the limitations of the existing directory solutions and ways in which this test domain is a good match to the semantic portals approach see the requirements document [⊣ REQUIREMENTS].

The demonstrator was developed using an iterative development approach. We were fortunate in being able to use data from an earlier publication *Who's who in the Environment* [⊣ WWITE] and are grateful to the UK Environment Council for supporting us in that. By taking a subset of that data, manually annotating and updating it and converting it to RDF we were able to build a test database that could be used for development purposes. We then built a first implementation of the proposed portal based on this dataset and used it in discussions with interested parties such as the Environment Council and the Natural History Museum. Feedback from those discussions led to revisions to the user interface, the classification schemes and the underlying software. The complete portal including data entry and harvesting support was then developed. The aesthetic appearance of the final demonstrator was greatly aided by graphic design input from Ben Joyner of ILRT to whom we are grateful. A second small scale testing process was undertaken with sample end users and data providers which led to feedback on detailed user interface and thesaurus issues leading to the final deployed system.

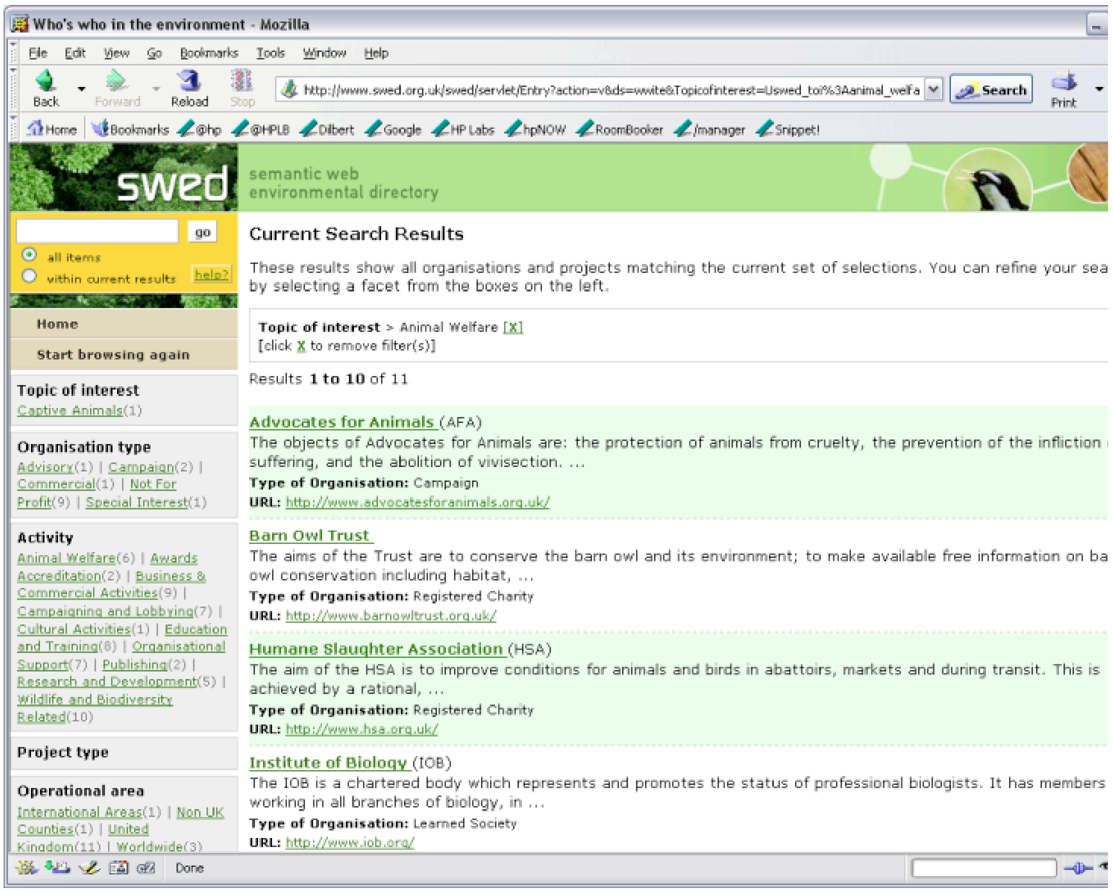
3.3 The SWED interface and functionality

The SWED interface allows users to search the aggregated pool of organization descriptions and to view the detailed description of the organizations found.

The search makes use of the ontologies and thesauri developed for the domain. The interface provides a number of *facets* along which the organizations are grouped. In the deployed SWED demonstrator we provide facets to represent the type of organization or project, its topic of interest, the activities it engages in and the geographical range of its operations. Each of these facets is described by a hierarchical thesaurus and the interface allows the searcher to select concepts from each facet and iteratively refine the search by further narrowing the facet or adding constraints from other facets. The current state of the search is displayed as a search *trail* to make it easy for the user to understand where they are and to remove constraints from the search. This *faceted browsing* approach is a standard interface technique in digital libraries. Our particular approach to this was particularly inspired by the Flamenco research project [⊣ FLAMENCO].

We also support free text search over the terms in the organization descriptions. The free text search supports boolean queries and queries restricted to particular property fields. A text search is treated as another facet constraint so that it can be displayed in the trail and combined with the constraints from the hierarchical facets.

The screen shot below shows the browse result screen which demonstrates each of these features.


Who's who in the environment - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.swed.org.uk/swed/servlet/Entry?action=v&ds=wwite&Topicofinterest=Uswed_toi%3Aanimal_welfa
Search Print
Home Bookmarks @hp @HPLB Dilbert Google HP Labs hpNOW RoomBooker /manager Snippet!
swed
semantic web environmental directory
go
all items
within current results
help?
Home
Start browsing again
Done

Topic of interest
Captive Animals(1)

Organisation type
Advisory(1) | Campaign(2) | Commercial(1) | Not For Profit(9) | Special Interest(1)

Activity
Animal Welfare(6) | Awards Accreditation(2) | Business & Commercial Activities(9) | Campaigning and Lobbying(7) | Cultural Activities(1) | Education and Training(8) | Organisational Support(7) | Publishing(2) | Research and Development(5) | Wildlife and Biodiversity Related(10)

Project type

Operational area
International Areas(1) | Non UK Counties(1) | United Kingdom(11) | Worldwide(3)

Current Search Results

These results show all organisations and projects matching the current set of selections. You can refine your sea by selecting a facet from the boxes on the left.

Topic of interest > Animal Welfare [X]
[click X to remove filter(s)]

Results **1 to 10** of 11

Advocates for Animals (AFA)
The objects of Advocates for Animals are: the protection of animals from cruelty, the prevention of the infliction suffering, and the abolition of vivisection. ...
Type of Organisation: Campaign
URL: http://www.advocatesforanimals.org.uk/

Barn Owl Trust
The aims of the Trust are to conserve the barn owl and its environment; to make available free information on ba owl conservation including habitat, ...
Type of Organisation: Registered Charity
URL: http://www.barnowltrust.org.uk/

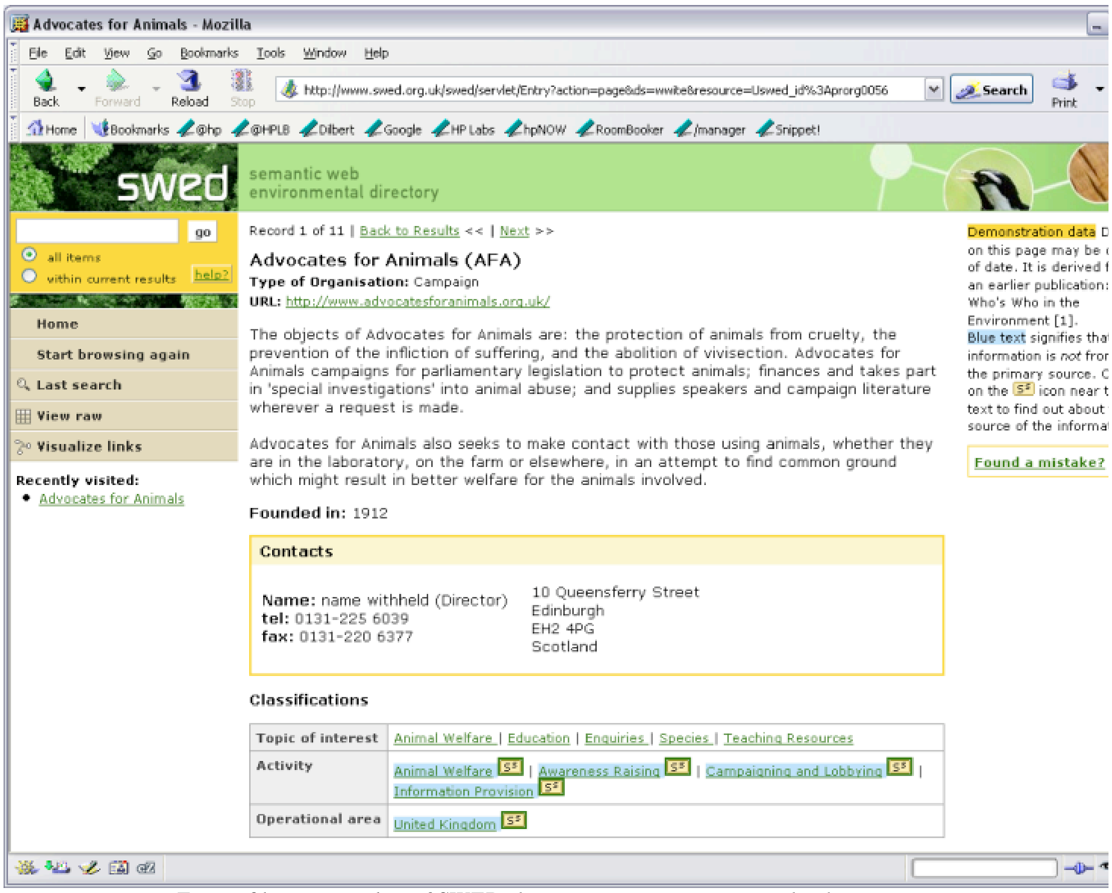
Humane Slaughter Association (HSA)
The aim of the HSA is to improve conditions for animals and birds in abattoirs, markets and during transit. This is achieved by a rational, ...
Type of Organisation: Registered Charity
URL: http://www.hsa.org.uk/

Institute of Biology (IOB)
The IOB is a chartered body which represents and promotes the status of professional biologists. It has members working in all branches of biology, in ...
Type of Organisation: Learned Society
URL: http://www.iob.org/

Figure 2a - screen shot of SWED demonstrator results page

When an organization is selected a web displaying the information on it, derived from the RDF description, is shown. This includes the classification of the organization and any relational links with other organizations as well as textual descriptions.

A particular challenge with a decentralized portal is how we convey to the user where the information originated from. Some organization descriptions come from historical or third party sources rather than the organisation themselves. A single organisation display page might include a definitive description by the organization itself coupled to classifications and links added by third parties. The approach we chose was to make the source of the overall page visible to the user, to high pages drawn from historical (and possibly out of date sources) and to highlight classification and link data on a page that is drawn from a source other than the main page source. This is illustrated in the screen shot below.


Advocates for Animals - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
Search
Print
Home Bookmarks @hp @HPLB Dilbert Google HP Labs hpNOW RoomBooker /manager Snippet!
swed
semantic web environmental directory
go
all items
within current results
help?
Home
Start browsing again
Last search
View raw
Visualize links

http://www.swed.org.uk/swed/servlet/Entry?action=page&ds=wwite&resource=Uswed_id%3Aprorg0056

Recently visited:
- Advocates for Animals

Record 1 of 11 | Back to Results << | Next >>

Advocates for Animals (AFA)
Type of Organisation: Campaign
URL: http://www.advocatesforanimals.org.uk/

The objects of Advocates for Animals are: the protection of animals from cruelty, the prevention of the infliction of suffering, and the abolition of vivisection. Advocates for Animals campaigns for parliamentary legislation to protect animals; finances and takes part in 'special investigations' into animal abuse; and supplies speakers and campaign literature wherever a request is made.

Advocates for Animals also seeks to make contact with those using animals, whether they are in the laboratory, on the farm or elsewhere, in an attempt to find common ground which might result in better welfare for the animals involved.

Founded in: 1912

Contacts

Name: name withheld (Director)
tel: 0131-225 6039
fax: 0131-220 6377

10 Queensferry Street
Edinburgh
EH2 4PG
Scotland

Classifications

Topic of interest	Animal Welfare | Education | Enquiries | Species | Teaching Resources
Activity	Animal Welfare S² | Awareness Raising S² | Campaigning and Lobbying S² | Information Provision S²
Operational area	United Kingdom S²

Demonstration data D
on this page may be
of date. It is derived
an earlier publication:
Who's Who in the
Environment [1].
Blue text signifies tha
information is *not* fro
the primary source. C
on the S² icon near t
text to find out about
source of the informa

Found a mistake?

Figure 2b - screen shot of SWED demonstrator organisation display page

The live demonstrator can be visited at: http://www.swed.org.uk.

In the next sections we describe the implementation of the demonstrator, starting with the domain ontology and associated thesauri and then moving on to describe the software components. The software components are designed to be generic, customizable components applicable to other similar semantic portals. Finally we discuss the deployment and evaluation experience, the current status and future plans.

3 The domain model

3.1 Overview

The SWED demonstrator needs to describe the following types of entity:

- organizations (name, descriptions etc.)
- projects (name, descriptions etc.)
- contact information (people, addresses, phone numbers, emails)
- classifications (type of organization, their interests, activities, geographic area)
- relationships between organizations

In order to represent this information in semantic web form we needed to develop or select appropriate vocabularies, ontologies or thesauri. Even though this is really quite a simple domain the choice of modeling approach to use was not always obvious. In the event we chose to use an OWL ontology [OWL] to represent the core structure of the organization and project descriptions and informal thesauri (represented using SKOS [SKOS]) for the majority of the classification schemes. We discuss some of the tradeoffs involved in these choices below.

A diagrammatic illustration of this hybrid structure is shown in the figure below:

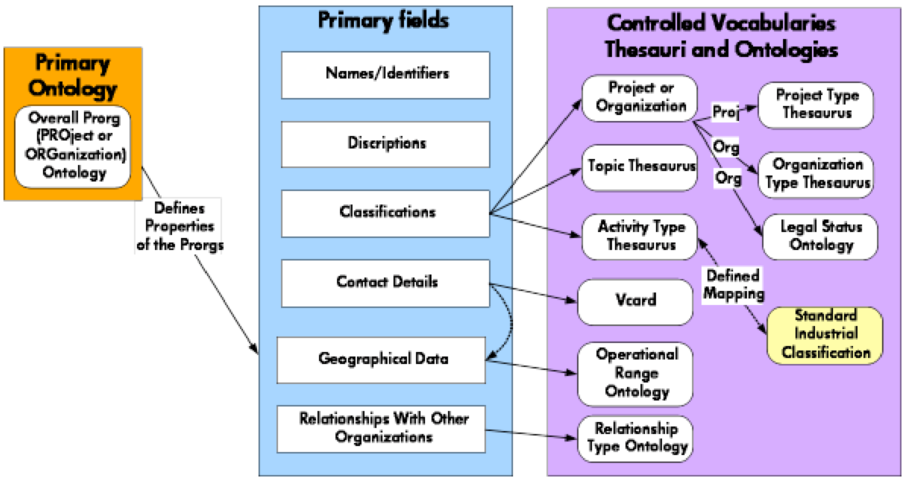

Primary Ontology
Overall Prorg (PROject or ORGanization) Ontology
Defines Properties of the Prorgs
Primary fields
Names/Identifiers
Discriptions
Classifications
Contact Details
Geographical Data
Relationships With Other Organizations
Controlled Vocabularies Thesauri and Ontologies
Project or Organization
Proj
Project Type Thesaurus
Org
Org
Organization Type Thesaurus
Topic Thesaurus
Legal Status Ontology
Activity Type Thesaurus
Defined Mapping
Vcard
Standard Industrial Classification
Operational Range Ontology
Relationship Type Ontology

Figure 3- illustration of the domain modeling components

3.2 The organization ontology

A central requirement of the system is that it provides a means of describing the core set of properties of organisations, parts of organisations and projects, that would be required for a directory entry. Our approach was to begin by identifying what properties would be necessary (or desirable) and then seek existing metadata standards/ontologies that would meet those requirements.
After reviewing existing paper and web based directories we identified a set of core properties:

- Organisation name
- Previous name
- Acronym
- Primary URL
- Logo (of organisation/project)
- Contact details
  - Contact person
  - Position
  - Care of (Organisation)
  - Postal Address
    - Building Name
    - Street
    - Locality
    - Region
    - Postcode
    - Country
  - Contact telephone
  - Contact fax
  - e-mail
- Description
- Classification(s) (e.g. topics of interest, areas of activity, geographic operational area)

And in addition a number of properties that, while not 'core' to a directory entry would be useful, e.g.

- Alternate URL
- Year formed
- Relationships with other organisations and projects (e.g. 'part of', 'project of', 'member of')
- Services offered
- Publications (e.g. regular journals as well as reports, etc.)
- Volunteering opportunities

A review of existing ontologies offered no obvious candidates for direct use. Where widely used standards existed (e.g. VCARD for handling contact details and addresses) they were incomplete or it was not clear how best to use them in the context of SWED (e.g. how to use VCARD for organisations raises a number of issues, see similar discussions [VCARD_DISCUSSION]). Where other smaller scale projects had defined aspects of ontologies that deal with organisations (e.g. AKTiveSpace [AKTIVESPACE]) they were generally very specific (e.g. AKTiveSpace focus on 'Educational-Organization-Unit's rather than a generic 'organisation'). In no case did we find an ontology that provided the range of properties that we required. In addition in no case did we find ontologies that described projects as well as organisations. However we acknowledged that such an ontology may well already exist or could be created by combining existing ontologies.

Given our initial findings and the pilot nature of the project, we decided to create our own ontology of organisations and projects. We did this in such a way as to allow properties to be added/refined easily to help make the system as extensible as possible.

The core ontology is based on the concept of a 'prorg' (i.e. a contraction of project/organisation) prorgs have a property 'prorg_type' that in the demonstration system can have values of 'organisation', 'part_of_organisation' and 'project'. This allows us to have a generic high level ontology for all 'project/organisations' that can be refined/extended for different sub-types.

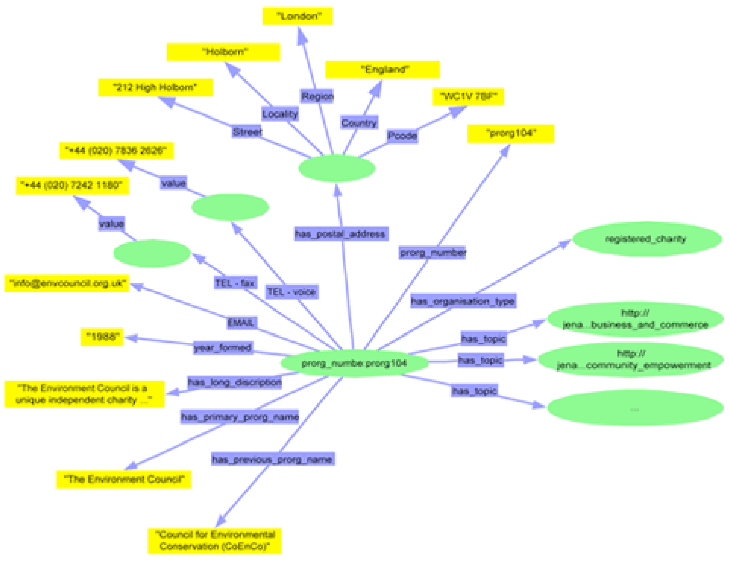


Figure 4 - simplified graphical view of an example of SWED RDF file

⊣ Figure 4 above illustrates the high level structure of the '⊣ prorg' ontology. As contact information was the most basic of information and most likely to be widely reused we decided to use the vCard format for contact details as far as possible - however there were limitations. For example, in vCard there is no 'care of' field which was required by large numbers of organisations and projects. There exists an W3C Note [⊣ VCARD] that describes a means of Representing vCard Objects in RDF/XML. We made use of core terms from that schema, though we avoided that schema's approach of typing of contact details using `rdf:value` as cumbersome and unnecessary in this context.

One design pattern we used repeatedly was to use subproperty hierarchies to provide extension points and group related properties together. For example, we defined a general `has_contact_details` property that groups the vCard and other properties together. This was effective in making the ontology more transparent.

Organisations are identified by their primary URL using OWL inverse functional property (IFP). This means that we can use primary URLs as a means of joining data about the same '⊣ prorg'. We use this explicitly in the relations property where '⊣ prorgs' can define relationships with other '⊣ prorgs', each specific relation e.g. project_of is a sub-property of a single 'has_relation' property, allowing for the arbitrarily extension in the types of relationship that can be defined.

It quickly became during initial discussions with more smaller/local organisations and associated network organisations that many organisations do not have web sites, or even web pages hosted on another organisations/projects web site. Indeed in many cases they do not have e-mail contact addresses. However, we were able to overcome this by using the `mailto:` and `tel:` URI schemes to represent the contact information and use that as the primary URL for organizations that lack their own web presence.

3.3 The classification thesauri

The classification thesauri/ontology are central to the concept behind SWED, the ability to locate organisations/projects using different facets of their work e.g. the topics they are interested in, the things that they do, the geographical area that they operate within, etc. were felt to be a valuable primary means to navigate the information space. This is reflected in nearly all existing directories that categorize entries under facets such as location, and topic of interest. Specific examples of faceted browse that use existing technologies include the voluntary organisations directory based at VOSCUR in Bristol [⊣ VOSCUR], and searchable directories using traditional search within particular data fields (facets) e.g. Envirolink, based in the East of England [⊣ ENVIROLINK].

Our goal was to develop set of facets that would allow users to refine their search/browse to locate the organisations/projects relevant to their needs, as effectively as possible. Each term/concept in the thesauri could have a scope note in order to help users and those entering the data for their organisation/project to choose the appropriate terms.

After reviewing existing directories (including the Who's Who in the Environment Directory) we decided on a small number of core facets - that would provide effective refinement using the faceted browse interface.

One generic and continually problematic issue was balancing the *size/complexity* and *specificity* of the classification schemes. The more complex and large the thesauri the harder it is for users (and cataloguers) to find the most appropriate terms. We did not have the development time to create a complex thesauri search/browse system and the user interface design of such systems are problematic. Our chosen interface approach (faceted browse) works most effectively with relatively narrow and deep taxonomies, so that there are never too many terms at one level to make display impractical (certainly the case with many larger taxonomies such as the Library of Congress Subject Headings). We therefore decided to attempt to create/choose our pilot thesauri and ontology to match that requirement and keep the number of terms at any one level relatively small (e.g. less than 100 terms). Given the generalist nature of the SWED project (i.e. it has a scope of all environmental organisations/projects) we decided to attempt to make/choose the thesauri to be at a relatively low level of specificity. This seemed sensible as if specialist directories were developed from the core SWED data they could extend or create their own specialist classification schemes to meet their needs. In order to make terms easier to locate for users we also chose to use multi-hierachical structures (i.e. a concept can have more than one broader concept), early preliminary testing seemed to demonstrate that strict single hierarchies made it much harder for users to locate terms below the first level.

Organization type - we originally worked to develop a basic but formal 'legal status' ontology for organisations. In principle this would in most cases be easy for organisations to classify themselves as (in general) each would only have one legal status e.g. public_limited_company, registered_charity, etc. And in parallel a more "colloquial" version that would include terms that while not tightly legally defined are widely used (e.g. voluntary organisation, network organisation) and are more likely to be valuable to end users than legal categories, which are not necessarily intuitive.

The potential offered by the formal legal_status ontology is that offers defined constraints for the ontology. The properties of an organisation will vary depending on the legal status of an organisation e.g. a registered_charity will have an registered charity number, while other types of organisation will not and a limited company will have a registered office. It would also allow the pre-population of some other properties, e.g. some terms within in the types of activity and more colloquial organisation type categorization. In principle (and the longer term) it would allow much richer integration with 3rd party sources such as the register of charities of England and Wales [┤CC] that provides very rich information on specific types of organisation.

However during development it was felt that there was not sufficient development time to implement a sufficiently rigorous legal status ontology and associated mappings to specific properties given the great complexity of the practicalities of the concept of legal status, and that the more more colloquial version would actually be more useful to users. We therefore decided to drop the legal status ontology and focus on the more colloquial organisation_type thesaurus. This was developed from the original data in the Who's Who in the Environment categories and added to, to widen the scope to focus on all likely types of organisation, e.g. including commercial sector organisations excluded from the Who's Who directory.

Project type - This mirrors the organisation_type facet for organisations. We could not locate any existing extensive classification system for projects. We therefore decided to create our own for the prototype based on various categorizations of projects across government, academic, voluntary and commercial sector. This was seen very much as a first pass attempt that would be refined if the project is taken forward. There are clearly issues with the current version, e.g. the concept of pilot project and its sub-concepts might be thought of as a distinct facet of a project when compared with other 'types' in the thesauri, such as research, campaign project.

Topic of interest - i.e. the things organisations/projects are interested in. This is the facet that initially seemed to be most widely used in existing directories. However under closer inspection most directories used a facet/element such as 'keyword' to cover both the topics of interest of an organisation and the things that the organisation does. This caused some issues in understanding the semantics of the categorization e.g. an organisation might be classified under the keyword education because it was *interested in* education, but didn't actually provide educational opportunities and vice versa. We therefore created two separate facets 'topics of interest' and 'types of activity' (see below), to make this distinction clearer.

We investigated the possibility of using existing classification schemes that have been developed for the environmental sector. These included GEMET - GEneral Multilingual Environmental Thesaurus [┤GEMET], the Biocomplexity Thesaurus [┤BIOTHES] and others. We also received feedback from the Environmental Thesaurus and Terminology Workshop in Geneva [┤ETT] via a SWAD-E colleague, where a number of other thesauri were detailed. However all of these were larger, more complex and narrower in focus than we required e.g. GEMET has wide coverage but is designed for classifying documents rather than organisations. We therefore decided to start with the existing Who's Who in the Environment index classification as our starting point and as with organisation type we extended it to cover a wider range of organisations/projects.

Activity type - As discussed in the previous section the activity type facet provides a classification for the types of activity that organisations/projects do (i.e. their activities). Initial work identified the SIC (Standard Industrial Classification) systems that are used across the world to classify 'economic activity' for statistical/monitoring purposes. Many countries have their own version of a SIC but all are similar to the International SIC [┤ ISIC] maintained by the United Nations. However the SICs are complex and in many cases counter intuitive for users not used to the system and it became clear that a simplified (environmentally focused), user friendly version would needed. We therefore decided to take the ISIC as a basis and develop a more focused and user friendly thesaurus, that could be mapped to the ISIC terms, and so retain interoperability with other systems that use ISIC.

Geographic operational area - The intention of this facet was to provide a means for users to locate organisations that operated in particular geographical areas, e.g. the area in which they lived or conducted research. However there are many different ways of dividing up geographic regions e.g. in the UK political, post code, health authority, national parks, travel to work area, and different organisations operate within those different boundaries. To add to the complication, many organisations were originally created to operate in regions that no longer officially exist e.g. the counties of Middlesex and Avon in the UK.

To overcome this it seemed sensible to make available all likely geographical classifications and allow those entering the information to choose the most appropriate to them. And ideally have the system map queries (via GIS representation of the regions) from one scheme (e.g. national parks) onto all others (e.g. UK county regions) so that when a user made a query they could (given an appropriate interface) see all the organisations/projects that were in areas that overlapped with the chosen region . Such a system would be complex and beyond the scope of the project - however the principle is well understood and systems already exist that make use of such facilities on the web e.g. UK GIGateway [┤ GIG].

For the demonstration we decided to implement a basic version of the same approach providing a number of different geographic classification schemes e.g. national parks and uk administrative regions; but did not implement any mapping between the regions. So the organisations/projects themselves add all the categories that they feel are relevant to their organisation.

We decided to implement the approach using a simple 'contained_within'/'contains' property to represent the hierarchical relationships, with a single root 'operational_area' with specific geographies e.g. UK (administrative areas), UK dependencies, Non-UK countries and worldwide. Worldwide has no sub-regions. Originally this was the root, however in many organisations have operational ranges that are technically worldwide but would also have specific geographical foci (e.g. the UK or particular countries). We therefore decided to overcome this issue with in the demonstration system by making worldwide an independent 'region', although clearly this is an unsatisfactory solution in terms of ontological modelling, and requires further research in order to identify a more suitable ontological structure. Exploring those used by other systems such as the Getty Thesaurus of Geographic Names [┤ GETTY] would be helpful.

The browsing of this facet is problematic as it is not clear how users will expect a hierarchical browse of regions to work e.g. if they enter 'Bristol' do they expect to see all the organisations that operate in 'Bristol' including *all* those that operate in regions higher in the hierarchy or those that operate in Bristol and all those in sub-regions of Bristol? This issue needs more feedback from user studies. Our very limited user feedback as of writing indicates that the latter is more intuitive and this is what has been implemented.

3.4 Lessons learnt - domain modelling

a. Context sensitive decisions - navigation versus rich representation

The appropriate structure and modelling detail for an ontology or thesaurus is dependent upon the context of use. In the case of the SWED demonstrator the classification schemes such as topic of interest are used in several contexts.

First there is the context of any end users searching the portal for organizations use it for navigation. In this situation a simple hierarchy (not too broad at each level) is ideal. Having overlap of terms or several paths to the same classification is not a problem since (we believe) these users are primarily interested in recall (finding relevant organizations) and are prepared to trade off some precision to achieve this.

Then there is the context of a representative of an organisation using the thesaurus for data entry. This situation is similar to that of an end user search but in this context the user would prefer each category be distinct and only on a single path so they can be sure of selecting the right one, the redundancy and overlap that would help searchers achieve high recall is an overhead for data creators.

The third context occurs when third parties merge the data with other data sources. In that context it is ideal if the classification scheme is as rich and formal as possible and based upon wider standards. Such schemes are typically much larger and harder to navigate for our two main classes of users. Our experience (see above) with attempting to use the SIC classification to represent organization activities is a good case in point. In that case we were able to develop a small navigable thesaurus for our own use but link the terms to corresponding SIC codes to facilitate future merging.

Thus one lesson is that one should take the main contexts of use into account when developing a knowledge organization scheme. This is hardly a novel observation but when thinking about ontologies and semantic web it is easy to focus on the requirements of precision and data integration to the exclusion of the requirements for end user navigation. The SWED demonstrator illustrates three techniques for compromising between the different tradeoffs. Firstly, our basic structure of a core ontology with associated classification thesauri (linked by properties identified using a property hierarchy) seems likely to be a reusable design pattern. Secondly, by supporting multiple classification schemes and faceted search we make it possible to have overlapping classification schemes with different tradeoffs side-by-side and let end users select those most appropriate to the task. Thirdly, the technique of providing a simple navigation oriented scheme for user navigation but anchoring the concepts in larger, cross domain, thesauri is a common and important semantic web design pattern.

b. **Navigation issues - UI metaphor needed**

Despite efforts to keep our thesauri small and suited to our navigation tasks initial feedback indicated that it could still be hard to find the right term in the thesaurus, especially during data entry. Our choice of UI tool (tree based views) is reasonable but for large trees then choosing which branch to expand, or seeing the wood-for-the-trees when lots of branches are expanded, remains challenging. Providing search over thesauri terms is a partial solution for very large thesauri but ideally would need to be coupled to some means for viewing nearby (ideally, semantically nearby) terms. There is an existing body of research in search and classification user interface using thesauri; within the scope of this project we have not been able to actively explore the many alternative UI metaphors. We simply note this as an open issue and an area in need of either better user interface solutions or more accessible recommendations of best practice based on current UI research.

This is a user interface lesson rather a domain modelling lesson but we mention it here due to its strong relationship to the above lesson on design-for-context-of-use.

c. **Lack of formal basis for most of the classification schemes**

When modelling the real world using an ontology we are typically imposing fixed discrete categories, relationships and constraints on concepts that are in fact continuous, overlapping and context dependent. Even seemingly fundamental dichotomies such as male/female are not simple (in some contexts one may need to account for notions of mental rather than physical gender, for hermaphroditism, surgical alteration, asexual single celled organisms etc.).

Initially we had expected to make stronger use of formal ontologies for the SWED classifications but in practice the bulk of our classification schemes are informal hierarchies of concepts which are suited to user navigation but not necessary anchored in fundamental distinctions in the world. This is partly due to our goals to support human retrieval rather than formal representation but largely due to there being few aspects of the domain which divide unambiguously into discrete concepts.

The one place where there was basis for a more a more structured classification scheme was *organization type*. The legal framework of the country (the UK in our case) provides reasonably fixed definitions for types of organizations, their relationships and attributes (for example that a registered charity will have a charity number). Though even here the need for intuitive user navigation (see (a) above) made it more appropriate to use a colloquial thesaurus and the differences between legal systems meant that any strong constraints that we might impose would be likely to be violated as the application expands to include information from other jurisdictions.

d. **Modelling open ended schemas**

A significant strength of the RDF [RDF] data model is its ability to support open extensibility of data. The combination of the open world assumption and the property-centric nature of RDFS [RDFS] vocabularies means it is possible to define new properties and add them to existing data "objects" without difficulty. This makes is very easy to extend and enrich data models over time, as well as to merge disparate models, and is a key benefit of the semantic portals approach that we wished to exploit. However the existing representations make it very hard to construct schemas which indicate what properties might be expected for a class without over constraining them. Difficulties which this arose surprisingly often in the design of the core Organization ontology.

To take a specific example consider the case of representing an email address for an organization. We wished to reuse the existing `vcard:EMAIL` element from the vcard schema [VCARD] (though in fact we did not use the rdf:value convention suggested in the note). This is defined to have an unconstrained domain, it could be applied to any class of resource, including our prorg class without problems. We do not require an organization to have an email address nor constrain it to a single address. So in fact there are no schema declarations to add. We don't have any cardinality constraints to express, the domain of `vcard:EMAIL` is already compatible, there is nothing to do. However, this means that the user of the schema has no idea than an email address might be permitted and that we would prefer them to use `vcard:EMAIL` to capture it, other than natural language documentation. There is no way to express the notion that a class has a expectation of a property value but the property is optional. The nearest one might do would be to add a domain declaration say that `vcard:EMAIL` has domain `swed:prorg` but this is clearly incorrect (all other uses of `vcard:EMAIL` would then incorrectly be deduced to refer to `swed:prorgs`). In the event we adopted an incomplete solution. We defined a parent property, specific to `swed:prorg` as a super property of such external properties. In this case we used `swed:has_contact_details` (and didn't introduce an additional `swed:has_email` subproperty but we could have done). We then defined `swed:has_contact_details` to be a super-property of `vcard:EMAIL`.This is still not correct because it does imply the same incorrect domain constraint but at least the intent of the schema is relatively clear and we are not directly altering the defintion of `vcard:EMAIL`.

This means there is explicit declaration within the schema that indicate and a vcard email address is one way of providing contact details, gives us a base property that the UI can use for finding such contact details but leaves the schema open. The contact details are still optional and a new property could be defined to fulfil the same role and added in as another subproperty of `swed:has_contact_details`.

e. **Tool support**

We found existing tool for support for generation of ontologies and thesauri largely sufficient to our needs within this demonstrator. We used the Protégé OWL plugin [PROTEGE-OWL] for both. In the case of Thesauri we developed them in Protégé using owl:Classes to represent concepts and subclass relationships to represent broaderTerm links and used a simple rule set expressed using the Jena [JENA] RuleMap tool to translate the OWL to SKOS format.

4 Architecture and implementation

4.1 Overview of the architecture

The overall architecture of the SWED demonstrator is illustrated in block diagram form in figure 5 below.

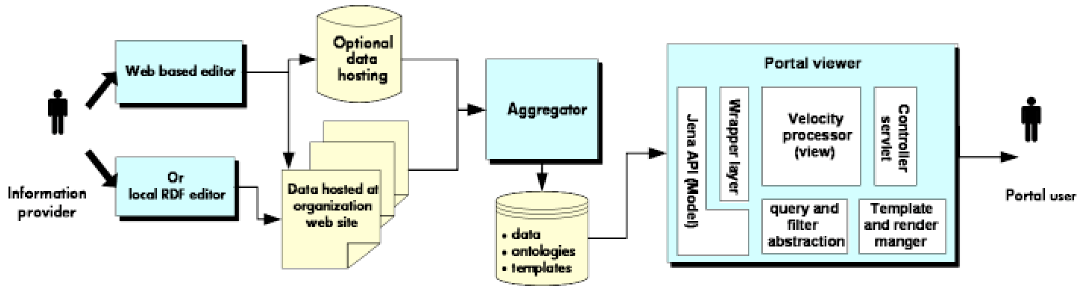


Figure 5- block architecture of the demonstrator

There are three primary tasks supported by the demonstrator (data creation, aggregation, viewing) each of which is implemented by a separate software module.

At the heart of the system is a data base containing the RDF data describing the organizations and their relationships, the ontologies and thesauri used in that data and the viewing templates that can be used to display the information in human readable form. We think of this as conceptually one data set, represent it in the software using a single `DataSource` abstraction but in practice only the RDF data is held in a true database and the templates and ontology information are stored in simple files.

The *portal viewer* component provides a web interface onto the database. It provides a faceted browse interface through which the set of organizations can by explored and the information on each organization can be displayed in readable web pages. This is a generic, template driven, component that could be used to be provide a range of different web interfaces onto RDF datasets.

The *aggregator* component scans known sources of relevant RDF data periodically and uploads any changed data sets to the portal database. This component currently runs within the same environment as the web portal (to simplify deployment and administration) but in principle could be a completely separate software tool.

The *data creation* component is a set of web applications to enable a provider organization to create their RDF description in the first place. The heart of this is a schema-driven web form that the information provider can fill in. This is coupled to a set of services to manage the overall data entry workflow - emailing the resulting RDF description to the provider to be hosted on their web site and registering the data once it has been published.

4.2 The web portal viewer

The task of the web portal viewer to take requests from a user, via a web browser, query the portal data to satisfy the requests and render the resulting information into a suitable HTML format for display back to the user.

We describe the high level structure and design approach for the portal here. A more detailed description of the software structure, configuration and administration is provided on the demonstrator web site [⊣ PORTAL DOCUMENTATION].

The portal is implemented as a single Java web application running in a servlet container. The demonstrator uses the Jakarta Tomcat [⊣ TOMCAT] servlet container as the host but other standards-compliant containers can be used. Within the web application the portal is architected using a web approximation to the Model-View-Controller (MVC) design pattern. We say "approximation" because we have used the philosophy behind the MVC pattern rather than followed precisely the set of Classes and Interfaces you would expect in a true MVC implementation. This is partly due to our use of a mixture of Java and scripted templates and partly because the web interaction model differs from the original MVC setting (web browsers *pull* information rather than web servers *pushing* it, so no need for the event propagation normally associated with MVC designs). Never the less, the philosophy behind the MVC design approach is relevant. First, by dispatching requests through a single controller we have a single place to intercept, log and reroute requests. Second, the separation of *model* from *view* allows us to have several different views onto the same data each of which reuses the same *model* abstractions.

4.2.1 Controller -

A single servlet acts as the controller component in MVC. Requests from the client web browser are directed to the controller servlet which extracts the action to be performed and the the associated parameters from the request URL. For the core browsing we only need to two actions - *view* and *page*.

The *view* action is used to display all resources that match a given set of search filters. The state of the search filter (both facet selections and free text search) is encoded in a standard form in the URL parameter. The action is translated into a search over the data for all resources that match the current filter and the returned results will be rendered using the View components appropriately. By convention within the portal a *view* action on an empty filter displays the top level browsing page allowing users to select an initial filter.

The *page* action is used to display a single resource as a web page. In the demonstration that resource will normally be the URI for an organization whose descriptive properties and relational links will be displayed. However, any resource URI can be specified and the View components will attempt to find appropriate templates for rendering that resource.

Several other actions are needed to display the same information in different views (e.g. show a *raw* view of the RDF data or a graph of the relationships between organizations) and to administer the portal. In a few cases these are sufficiently specialized to warrant their own servlet implementation but the majority of such actions are dispatched view the main controller servlet. The bulk of such requests are simply requests for different views over the same result data so we adopted a naming convention which allows the controller servlet to map the name of the request action onto the name of a *view* template that can satisfy the request. This allows the portal interface to be openly extended by just modifying the viewing templates.

A full description of how actions, data sources, resources and search filters are encoded within the URL request to the controller servlet is included in the [PORTAL DOCUMENTATION].

4.2.2 View -

The job of the *view* component of the portal is to take the set of RDF data retrieved from the *model* and render it for display.

A design goal was that it should be possible for developers to adapt the portal to display new data and change the look-and-feel and navigation structures without necessarily having to write Java code. To make this possible we chose to use a template-driven approach. To render the view for any given request a suitable display template is located and that is used to extract and render the individual data components.

For the template engine a number of choices were possible. For example, since we are working within a Java Servlet environment then Sun's JSP technology [JSP] would have been a possible choice. However, we had a couple of design requirements that affected the selection of template engine. First we wanted to support embedded templates. For example, we wanted it to be possible to produce templates for displaying address fields described using different ontologies and then, in a template displaying an organization, simply ask for the address to be displayed inline leaving it to the system to decide which type of address it is and so which embedded template to use. Second we wanted it to be possible for third parties to publish templates over the web, just as they might publish extension ontologies and additional data annotations, and have the portal be able to dynamically discover and reuse relevant templates.

To meet these requirements we chose the Jakarta Velocity template engine [VELOCITY]. This offers a simple and compact scripting language suited to the task of generating portal views. The programming model for Velocity is straightforward, templates are strings which can be passed as parameters, which greatly simplified dynamic retrieval and nesting of templates compared to the compiled approach of JSPs.

When the controller servlet receives a request to display a resource, or a set of search results matching some filter, it determines an appropriate template to use and hands it over to the Velocity engine to display a response using the selected template. The choice of the template to use depends on the action being performed, the type of the object being displayed and the data source configuration. Each portal data source can be configured to use a different set of templates for a given object type and view context. The set of viewing contexts is openly extensible. All of these parameters are defined in a single RDF configuration file which is loaded by the portal web application.

When a template is executing it has access to the request and the *model* data and can make recursive calls to the engine to render embedded objects using dynamically selected templates. This recursive approach makes it possible to write quite generic templates. For example, the top level portal pages for rendering a set of search results (in response to a *view* action) and a single resource page (in response to a *page* action) are generic and independent of the types of data being manipulated. When the generic templates need to display a summary search result or a full description of a resource they can render those through a recursive call to the rendering engine and the late binding will ensure that the appropriate subtemplates will be used for the type of data which has been found.

We also designed the generic templates to be modular so that it is possible to change the look and feel elements such as display headers and footers without altering the main templates.

The combination of a template-driven approach with dynamic binding of templates configured via an RDF-based specification has led to a very customizable solution. In fact the software is structured so that a single portal web application be used to view multiple *Data Sources* each with their own different sorts of data rendered using different templates and style sheets. For more details on the configuration options see the *Portal Customization Guide* in the [PORTAL DOCUMENTATION].

4.2.3 Model -

We implement the *model* part of the the MVC triad using a group of Java classes to provide an abstraction layer between the underlying data sources and the views which render the data.

As illustrated in ⊣ figure 3 above, the RDF data is all manipulated via the Jena [⊣ JENA] semantic web toolkit. This provides an abstract API for manipulating RDF and OWL data and includes multiple storage, query, inference, parsing and serialization tools.

The portal uses Jena Models to store and access the RDF data (*Model* is a very overloaded term, we use *Model* with a capital *M* to refer to the Jena API object and *model* with a small *m* to refer to the bundle of the wrapper classes which work along side the views and controllers in the portal implementation).

The portal can be configured to display data held in a Jena database or to load data from files and serve it from memory.

The portal design adds an additional abstraction layer between the Jena API and the view components. This abstraction layer itself can be thought of as two groups of classes - the first provides convenience wrappers around the relevant Jena RDF API objects the second adds additional abstractions which are specific to the portal application.

Convenience wrappers for resources

The portal software provides a set of convenience wrappers around the raw RDF API objects (for example encapsulating `RDFNodes` using `NodeWrapper` objects). These provide a place where utility functions can be added which simplify the scripting of the portal application. Typical examples of this functionality includes automatic ordering of statements based on lexical ordering of property names; determining the display name for an RDF resource (searches for `rdfs:labels` in the raw data and associated ontologies before falling back on qname forms) and truncating long text literals to simplify summary displays. The interface onto these wrapper objects is designed for ease of scripting at the expense of error checking. For example, we make it easy for the template scripts to address RDF properties by using qname strings (with the qname prefix taken from the set of XML prefixes defined in the bootstrap data files), this requires a dynamic lookup stage which could fail at runtime. This runtime error discovery is a typical consequence of scripting approaches.

Most of the functionality provided by these wrapper classes is straightforward and not relevant here (see the JavaDoc and documentation pages in [⊣ PORTAL DOCUMENTATION]). However, a couple of features (provenance, property enumeration) are less trivial and highlight issues in the portal design so we go into those in more detail later.

Application-specific abstractions

In addition to the RDF data itself there are several abstractions provided as part of the *model* layer which provide the structure of the faceted browse portal and simplify the scripting considerably.

Firstly, we need objects to directly represent the state of a portal search. The core idea of search in the portal is to divide the search space into a set of dimensions, called *facets*. Each facet specifies some property of the objects being searched over. This might be a simple keyword value or a hierarchical classification. We represent this using a `Facet` interface which defines the nature of the facet (name, type, properties used to represent it in RDF). The delivered software provides implementations of `Facet` for hierarchical category values and alphabetically ordered string labels but other implementations are perfectly possible. The specific `Facet` definitions for a given portal instance can be defined in the RDF configuration file. In addition we need to represent the state of a current search, which we do using a `FilterState` object which itself is a collection of `FacetStates` (each representing a selection of a specific refinement for a specific `Facet`).

This choice of abstractions makes the data flow very straightforward. The `FilterState` class includes methods to translate between URL request strings and `FilterStates` so the controller servlet just needs to take the incoming request URL and instantiate the corresponding `FilterState`. This gives a convenient point for caching - see below. A `FilterState` encapsulates the methods needed to find all matches to the state and to enumerate the possible filter refinements and count the number of matches to each of those refinements. This means that the scripts to implement the faceted browse are reduced to simple manipulation of the `FilterState` objects.

In addition we encapsulate the configuration of the portal as an explicit `DataSource` object. This makes it easy for a single instance of the portal web application to run multiple data sources concurrently. The `DataSource` object provides access to the configured ontology and instance data via a separate `DataStore` abstraction. The `DataStore` encapsulates the combination of different Jena Models which make up the portal data (providing access to the instance data and ontology information both separately and combined) and provides a single point for the inference support hooks (see below).

Caching

One important function of the portal abstraction layer is cache control. As noted earlier the portal can be configured to serve its data out of a database. This can be quite expensive. To display a faceted browse page showing a set of search results we need to:

- query the database for all resources which match the search
- for each matching resource retrieve enough of the descriptive properties to render a descriptive

summary
- find all remaining search refinements and for each of these count the number of resources that would match if that refinement were to be added to the search

The basic query to find all matching resources is simple and cheap.

Querying each resource for its descriptive properties increases in cost as the number of results increase. We mitigate this cost by limiting the number of results displayed per page (in the template scripts so it can easily be altered by developers) and by caching these properties when they are retrieved. We perform this caching within the wrapper layer. When a wrapper for a given resource is required we consult a cache to see if the wrapper has already been created then in the wrapper object we cache all property values for that resource after the first time they have been requested.

The refinement counting is by far the most expensive part of the faceted browse user interface. The demonstration portal has six facets each of which can have 10's of options at each stage of refinement. Thus displaying the refinement count can easily require on the order of 100 queries and this cost is the main limitation on portal responsiveness. We address this in several ways.

First, and most trivial, the refinement counts can be switched on or off by setting a single control variable in the viewing script. This enables portal instances to completely bypass this step if desired.

Secondly, when the data is loaded into the database we have the option to precompute and cache any set of refinement counts. It is not feasible to precompute all refinement sets because there are a combinatorial number of refinement sets. We do precompute the first level refinement counts for each facet (i.e. the number of items which match each first level concept below the root for each facet tree) but not the combinations of facet selections.

Thirdly, we maintain a dynamic cache which maps portal requests (as encoded in the request URL) to filter state objects (see above) which include the refinement counts for that state. In that way once a page has been generated then rebuilding it will be much faster for as long as the request state is in the cache. Setting this cache size allows a tradeoff of performance against storage costs.

Finally, it is possible to wrap a caching proxy server around the whole web application and cache the rendered HTML pages rather than the internal data structures - though this is outside of the scope of the portal web application itself.

Inference

In order to support the portal functionality we make use of Jena's inference capabilities. There are three main groups of inferences which are required.

Firstly, several of the facets in the SWED demonstrator make use of the SKOS thesaurus format to define hierarchies of concepts. When a search is made for matches against a specific concept (e.g. "topic_of_interest=Animal_Welfare") then we also want to retrieve all the organizations classified under more specialized (i.e. "narrower" concepts such as "topic_of_interest=Captive_Animals"). We achieve this by using a set of Jena rules to precompute the closure of the hierarchy For this example the rule is:

```
(?P swed:has_topic ?T) -> (?P swed:has_topic_cl ?T) .
(?P swed:has_topic_CL ?T) (?T skos:broader ?B)
                        -> (?P swed:has_topic_CL ?B) .
```

where `swed:has_topic` is the property that relates an organization to one of its topic of interests and `swed:has_topic_CL` is the closed version of that relation that we are computing via the rule.

Secondly, the organization ontology makes use of some OWL constructs that we wish to use in the navigation. In particular it defines certain inter-organization relations to be `owl:inverseOf` each other. We use rules to precompute the entailments from these declarations.

Thirdly, when a new organization is defined it can declare relationships between it and other organizations in the directory. Whilst directory entries can have resource URIs they are not necessarily definitive or known to data providers. Instead the person doing the data entry defines the organization they with to refer to by using its primary_url. In the ontology we declare this to be an `owl:InverseFunctionalProperty` so that it uniquely defines the target organization. We support the merging of nodes based on owl:InverseFunctionalProperties at the time data is imported - an operation sometimes referred to as *smushing*. This is done using special case Java code.

The inference machinery is all packaged as part of the `DataStore` abstraction so that portal scripts or custom servlets just need to add new data sets to the store, the store will run the relevant closure rules and invoke the smushing utility. The set of closure rules to be used is also defined as part of the portal configuration file. This makes it easy for portal developers to tailor the inference usage to their requirements. Typically small specialized rulesets are used, rather than the generic RDFS and OWL rule sets from Jena, for performance reasons.

Provenance

The web portal application displays data which has been aggregated from multiple locations. In the case of the SWED demonstrator the source locations include bootstrap data files, RDF files published by individual organizations and third party data bases. When a page describing an organization is displayed the data may have come from the organization itself or some some third party description.

What's more the single organization page may include information from multiple sources. For example the main description data may have been published by the organization itself but the page might show additional relational links and classifications published by a third party. The SWED user interface needs to make it possible to see where this data has come from. In particular it must be possible for the UI to highlight individual properties on a description page which were published by someone other that the main page data publisher.

The provenance tracking is supported by utility functions associated with the abstraction objects outlined above. These make it possible for a template script:

- to determine the primary source of a resource description
- to determine the source of a specific property value and test if it is the same source as the primary description
- to retrieve some predefined descriptive properties of the source to enable display of provenance information to the web user

The core machinery for this is a Jena `MultiModel`. This is an experimental API interface that collects multiple Jena Models together, each associated with a source URL and allows both the assembled union and the individual source Models to be queried. Any statement retrieved from the union can be mapped to the URI of the source Model(s) from whence it came. This general interface has emerged as a requirement some several projects but the current implementation (which layers on top of any existing Jena Model implementations) was developed specifically for the Semantic Portals demonstrator and is itself a small but useful spin off of the demonstrator.

The rest of the wrapper objects provide convenient access to this generic API to simplify the scripting needed to link the provenance information to the user interface. The only nontrivial part of this is the notion of a "primary source" for a resource description. On its own a resource description is a collection of RDF statements about a given Resource and each statement could have its own source. We experimented with notions of majority voting for "primary source" but decided that was too cumbersome and unpredictable. Instead we have chosen the very simple approach whereby the portal configuration can define an RDF property that represents the primary description. In the case of the SWED demonstrator that might be the textual organization description property. The source of that primary property is taken to be the primary source of the overall resource description. This approach has limitations but it has the overriding benefit of being controllable and simple to comprehend.

The provenance API itself has no built in notions of trust. It assumes that the sources and source descriptions presented to it are correct and presents them to the UI scripts on demand. Trusted provenance is beyond the scope of this demonstrator but were it to be required it should, in any case, be supported by the aggregation layer not by the provenance access layer.

Property enumeration

Part of the semantic portals vision is that, over time, members of the community can enrich the data in the portal by adding new properties, relations and classifications to link to additional information. Ideally we want the web portal application to be able to display this additional information with minimal work. In particular one potential pitfall with our template scripting approach is that the relevant display template chooses which information to display in which order and so might miss new extension properties that weren't known about when the template was written. However, completely generic templates that display all properties in a uniform way tend not to produce legible or aesthetically pleasing displays.

The solution we adopted is a useful compromise between fixed and generic templates. We provided tools (as part of the wrapper classes) to enable template writers to iterate over groups of properties. If new extensions are placed in an appropriate property group then the template can be written so as to be able to display the extensions appropriately. If that is not possible then the templates will also need extending in order to correctly visualize the data extensions.

Property groups can be defined in several ways. The most obvious, and the only one actually exploited in the SWED demonstrator, is to use subPropertyOf hierarchies. For example, in the SWED demonstrator all inter-organisation links are defined to be sub-properties of a base `swed:has_relation` property; so the display template can locate and format all such relations by traversing the subProperty hierarchy. A second option is to label properties explicitly by defining a class (of properties) and tagging each relevant property as being a member of that class. The third option that we support is the use of namespaces, grouping all properties which come from the same namespace together.

The wrapper classes include convenience functions to make it easy for template scripts to iterate over these property groups. This enables templates to be more robust to data evolution than would be possible in a fixed template mapping approach.

4.2.4 Lessons learnt - web portal -

Overall there were no fundamental problems with building a functioning and very customizable web portal solution using existing tools, notably the Jena Semantic Web toolkit and Apache Jakarta tools - Tomcat and Velocity.

In terms of software implementation and design there are a few points to highlight. In a later section we will discuss the user interface lessons learnt from the SWED demonstrator itself.

Template scripting
: The approach we adopted of using a templating language with late binding of the templates (based on context and type of object to be displayed) was very successful. We are happy with the modularization it made possible. In particular it was possible to implement the faceted browse user interface within the template scripts in a way that is independent of the particular facets, facet types or object types known to the portal.

Model abstraction layer
: One consequence of the use of template scripting was the need for an adapter layer to simplify the scripting needed to drive the underlying RDF machinery. This worked well in practice. The abstractions that are specific to the portal application (such as `Facet` and `FilterState`) were particularly successful and would be reusable in other related applications. The wrapper objects which match the generic RDF API to the scripting language could be reused but we caution that their interfaces evolved according the needs of the demonstrator application rather than being based on a principled top down design. As we gain more experience with customizing the applying the web portal tool we expect the wrapper interfaces will be further generalized and improved.

RDF processing rules
: The use of a generic RDF rule processing language (Jena rules) was helpful at several points in the design. In terms of initial data and ontology preparation we used the rule processor to perform simple rewrites between formats (for example, from an OWL class hierarchy to a SKOS concept hierarchy). More significantly we made use of the rules to precompute inferences that enable the user interface to benefit from the domain ontology (concept inheritance, inverse properties). This gives quite a powerful customization hook in that a specific portal application just needs to specify an appropriate rules file to implement the required semantic entailments. The alternative would have been to assume a generic OWL processor but with rules we can express transforms or entailments outside of OWL (such as those used for SKOS).

Provenance and MultiModels
: One lesson from this project for tool builders is the need for richer representations of provenance information when merging multiple datasources. This is not a new observation and some tools use the notion of quads or context labels (e.g. [REDLAND]) to allow triples to be tagged with additional provenance information. The MultiModel API proposed for Jena appears to give the benefit of provenance tracking without constraining the implementation to be quad-based and, on the basis of our experience with this demonstrator, gives sufficient functionality.

Open ended templates
: We noted earlier to need for templates which are robust against evolution of the data. The solution we used, grouping properties into extensible collections, was an improvement over fixed templates and sufficient for SWED. However, there is more that could be done here. For example, it might be useful to enable a default template section which can display all other information on the given resource that hasn't been covered elsewhere in the template. This would enable templates to ensure that no data extensions are ignored even if they have to fall back on a vanilla property/value table approach to displaying the extensions.
: On the whole our approach here should be regarded as pragmatic engineering workaround rather than a definitive solution and we regard this area as one ripe for further investigation.

4.3 Aggregator

The second software component which makes up the semantic portals implementation is the aggregator (also called the *harvester* in some of the software documentation).

The task of the aggregator is to fetch RDF data from publishing sites and load it into the database so that the web portal can be used to browse it. In the case of the SWED demonstrator the RDF data is published by the individual organizations - they use the data creation tool (see next section) to generate a description file in RDF, publish that file on their web site and then notify the aggregator that it can be picked up. The aggregator will continue to poll the known sites periodically and if the data file has changed the database will be updated to reflect those changes. Old data is not preserved.

The aggregator is not, and should not be, a generic semantic web crawler. The aggregator is restricted to only poll relevant publishing sites. This is partly to limit the volume of data that is fetched and managed by the portal. More important, however, is that we need some control over the source of the data. We don't want spoof information, spam or other inappropriate content to be included in the aggregation. For the SWED demonstrator this is potentially sensitive so we adopted a "white list" rather than "black list" approach. That is we only poll for information from known and trusted sites rather than load from any discoverable sites which aren't explicitly blocked.

We adopted a simple state model for sites known to the aggregator. The set of states used is:

new
: This indicates an RDF source site that has been registered via the aggregator registration interface but which has not be validated by a portal administrator. In the SWED demonstrator such sites will polled even though they have not been validated but will be prominently displayed on the portal administration page to encourage the administrators to validate them. In more sensitive applications it might be more appropriate for *new* sites to not be polled.

known
: This indicates a site that has been registered and validated by an administrator as an acceptable site to poll.

trusted
: This indicates a known site which is also trusted to introduce additional sites to the portal. In a conventional RDF crawler then links from one RDF source to another are often expressed using the `rdfs:seeAlso` property. If a site is *trusted* then the aggregator will follow any embedded `rdfs:seeAlso` links in that site's data and treat sources found there as *known*. If a site is *known* but not *trusted* then site itself is polled but any `rdfs:seeAlso` links are ignored.

blocked
: Indicates a site which was registered but rejected by the administrator. Such sites will not be polled and their data should not be displayed in the portal.

The implementation of the aggregation service is quite straightforward. So much so that it is implemented as a part of the web portal component rather than being a completely separate application. A separate RDF database is maintained by the aggregator which describes each RDF source that the aggregator is aware of. The RDF description includes the state information (as outlined above), information to assist with polling (`lastModified` date stamp and an optional digest of the last loaded file contents) and descriptive text which is used to describe this source when displaying provenance information on the portal web pages. A set of administration pages (written using the same template engine as used for the rest of the web portal) enables an administrator to retrieve and change site information.

There only two notable technical challenges in the aggregator design and these have already been addressed earlier. The first is that the data needs to be enhanced with inference entailments before it is added to the main database. This done by running the same inference ruleset and "smushing" machinery described earlier. The second issue is that we need to be able to dynamically add and replace the RDF datasets from each source into the combined dataset queried by the web portal. This accomplished by the MultiModel implementation also described earlier.

Lessons learnt - aggregator -

For this application the aggregation problem is straightforward and there are few general lessons to be drawn from the software design and implementation issues. The most important issue to note is that of the trust model. The demonstrator has a very simple trust model which is adequate to the purpose but leaves the general problem of trust models for semantic portals open to further work.

For the purposes of the semantic portal we can break the notion of "trust" down into a number of finer grained issues:

Should the data provided be included within the portal at all?
: Here the issue is whether the provider is genuinely offering data relevant to the portal or whether this is irrelevant (which would simply use resources unnecessarily) or deliberately inappropriate data such as spam (which could undermine the utility of the portal and alienate users and providers alike). In the demonstrator this is addressed by only aggregating from "white list" sources and using a manual validation process to determine acceptable sources.
: Other applications may ingest data from a wider range of sources and may need to switch to a "black list" approach. Automatically estimating relevance of a genuine dataset might well be possible and an interesting research direction. For example, if the data has no classes or

properties in common with the ontologies known to the portal it might be deemed irrelevant. However, reliably detecting deliberately inappropriate content is not feasible.

For what queries should the data be regarded as definitive?

When resource description page is displayed how can the portal user be sure whether the data is definitive or not? For example, how can they distinguish between a description of the Environment Council by themselves and a description of them by a third party? The solution used in the demonstrator is to provide provenance information as part of the user interface. Thus information entered from historical or third party sources is shown as such. Resource annotations from a source other then the main source for that resource are highlighted. However, we are leaving it up to the portal user to decide, on the basis of the source, whether the information is trustworthy. We have provided early hooks for recording what organizations a given aggregated data source can be regarded as definitive for but have not linked that information to the user interface.

More sophisticated trust models are possible. In particular it would be useful to distinguish the topics and properties for which a source is trustworthy. We might trust an organization to describe itself and its topic of interests but might prefer to trust an independent review body to classify the activities the organization engages in. The challenge is as much in the user interface to convey this trust model to the portal user as it is in representing and acquiring the trust information in the first place.

Is the data we receive what the provider intended?

A common concern in more sensitive areas would be whether the data transfer path is secure. If we harvest data from a trusted source how do we prevent a man-in-the-middle attack modifying the data en route. This is a standard data security problem which is not specific to the semantic web and is not a serious concern in this application. Current cryptographic solutions can easily be applied to semantic web data. There is a technical issue in cryptographically signing an RDF model (since the same model can have multiple serializations and normalization of them is an NP-hard problem). However, for the purpose of the portal aggregator it would be sufficient to just sign the transmitted document not the data content. This would give an adequate audit trail in applications that required this.

Who has the power to alter and update any given data set?

Again this is a data security issue rather than a trust issue but is an important one. In the demonstrator this issue is solved by the decentralized design. The data providers own and publish their own data. They have complete control of it and changes to it. The portal merely aggregates that data and has no support for individual edits to the supplied data. Third parties can provide additional assertions that relate to a data source (in which case the provenance issues noted above come in to play) but can't remove assertions.

4.4 Data entry

The *data creation* component is a set of web applications to enable a provider organization to create their RDF description in the first place.

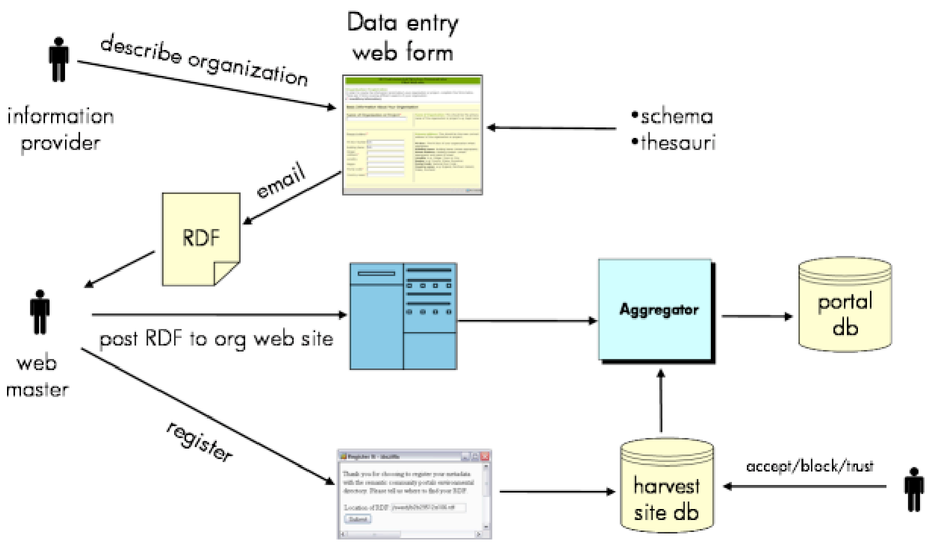


Figure 6 - input dataflow

Workflow -

Figure 6 shows the overall workflow. The intended usage can be described in these stages:

1. A user accesses a web form (see Appendix C) to create metadata for his/her prorg. This web form may be hosted locally to the user or remotely; in the demonstrator it is hosted on the same machine as the portal but this is not necessary. In fact, in contrast to most portals, the data creation is decoupled from the portal entirely.
2. Once the metadata has been created an email (see Appendix C) is sent to the user with:
   - the location whence the created RDF can be retrieved (this is usually hosted by the data entry server)
   - Instructions for hosting the RDF on the prorg website (step 3)
   - a link to a registration form (step 4)
3. The user downloads the RDF, and hosts it on the prorg website. The prorg now owns this data; it can protect it from change, provide links to it from its own website and can in principle update it or otherwise modify it. Aggregators can pick this up automatically if they scan the appropriate prorg website. Alternatively, users may explicitly notify one or more portals using the registration process (step 4).
4. Registration of prorg RDF is via a 2nd web form (see Appendix C). This web form accepts the location of the RDF (on the prorg website) and notifies one or more harvesters. In our demonstrator the harvester is implemented as part of the portal, and the registration process is implemented as part of the data creation application.
5. The portal harvester then polls the prorg website for its RDF. However the RDF is not 'trusted' until a portal administrator explicitly accepts it. This provides protection against spamming and other unsuitable portal content.

Design -

The design is centred around a schema-driven web form that the information provider can fill in. This was implemented as a web application in Java, using Tomcat [TOMCAT], JSP [JSP] and Jena [JENA] technologies. Some key design decisions are listed below:

- The three different prorg types (project, organisation, type of organisations) required different input forms; these were served by the same JSP (`type_entry.jsp`) which consulted a configuration file (in N3) to control the form. This is described in more detail below.
- The various web service functions (submit entry, send mail, registration form, registration complete) are handled by a single servlet which uses the request parameters to determine the appropriate action to take (and then dispatches the request to an appropriate JSP).
- After registration is complete, the RDF is stored locally for download by the organisation.
- Vocabularies are shared (copied from) the portal. There is no attempt made here to manage the co-ordination of divergent ontologies; it is expected that a single community (environmental prorgs in this case) will be happy to use a set of communal ontologies, certainly for the purposes of this portal.
- The presentation of UI controls for classification ontologies and for relationships was not straightforward; this will be discussed further below.
- Users were allowed to preview and to change their metadata after submission - this requires reloading the data creation form which is rather more involved than simply using the browser 'Back' functionality. Reload functionality is also discussed below

Configuration schema -

We designed a configuration schema to control the data entry form and associated functionality. The full schema can be found in the technical resources [TECH_RESOURCES], here we mention a number of the important features.

- The owning resource for an information item is called an `ItemType`, which contains a set of `FieldGroups`, each of which in turn contains a set of `Fields`. The `Fields` correspond to individual attributes (name, primary URL, contact name etc) and the `FieldGroups` enable us to group them (with, for example, a single background style). All the remaining design notes in this section pertain to `Fields` (usually) or `FieldGroups`.
- `Fields` have a `controlsProperty` attribute that links them to an `rdf:Property`. This may be straightforward (eg name, description) or it may be indirect (eg contact details, relationships). The attribute `hasValue` specifies whether that field should be interpreted as a literal or (RDF) resource.

- `Fields` and `FieldGroups` have `scopeNotes` associated with them. These are different to the scope notes of the `controlsProperty` target because they instruct the user how to use these fields in the context of the form. They should perhaps have been called `usageNotes`. For example, the scope note of `dc:title` is "A name given to the resource" whereas the usage note might be "This should be the primary name of the organisation e.g. legal name". (This example is illustrative: we didn't actually use `dc:title` in our metadata set).
- Both `Fields` and `FieldGroups` have an associated priority which enables them to be ordered in a consistent way. The algorithm used ordered all resources with a priority in priority order (lowest first), then all resources without a priority alphabetically by label (the label used was the `rdfs:label`, or the `dc:title`, or the local name, in that preference order).
- `Fields` may have mirror properties, which specify a URL parameter to set to the same value as that field. For example, when sending a mail to the SWED administrator, the servlet looks for a 'from' URL parameter, which is mirrored from the organisation's contact person. Conversely, `hasInitial` specifies a URL parameter from which a field gets its initial value.
- `Fields` may be hidden (`visible` is false) which enables us to set internal parameters (eg `hasProrgType`) using a `hasDefault` property (or even `hasInitial` for run time flexibility).
- Validators are used to correct certain inputs (eg add an "http://" if necessary to a primary URL).
- Classifications are specified using a `hasVocab` property which points to a SWED vocabulary file. This is explained in more detail below.
- Relationships also use `hasVocab`, but in their case the vocabulary is used to specify the relationship type (as `hasPart`, `partOf`). This is also described in more detail below.

UI for indirection -

Indirection is handled in one of two ways. Relationships use an `indirectProperty` attribute, which means the the value entered will not be the object of the property associated with the field, but will be indirected via a bNode and this `indirectProperty`. So, for example, a relationship with an indirectProperty of `:hasPrimaryURL` might be represented by the RDF

```
:thisOrganisation :partOf [:has_primary_url http://foo/bar ]
```

Note that since this is a relationship, the type of relationship (`:partOf`) is itself chosen by the user form a vocabulary (specified by `hasVocab`). This is described in more detail below.

An alternative method of representing indirection is using the `hasIndirect` property. So for example the a telephone number has a `controlsProperty` of `rdf:value` and a `hasIndirect` of `vcard:TEL`. This leads to the RDF of the form

```
:thisOrganisation vcard:TEL [rdf:value "123-456-789"]
```

This functionality, while important, also led to difficulties unambiguously identifying certain fields (eg telephone and fax) so we used `hasIndirectContext` to specify certain properties of this indirect node. Thus telephone and fax numbers can be disambiguated from the RDF:

```
:thisOrganisation vcard:TEL [rdf:value "123-456-789"; rdf:type vcard:fax ]
:thisOrganisation vcard:TEL [rdf:value "123-456-789"; rdf:type vcard:tel ]
```

There is another issue with indirection. Given two fields that specify indirect properties, should they be aggregated in the same bNode or left separate? For example, VCARD address properties are specified separately on a form but can be aggregated into the same bNode. On the other hand, `vcard:TEL` may pertain to voice or fax, and these should be separated into different bNodes (disambiguated by `rdf:type` as shown above). Hence the `isIndirectSeparate` property.

UI for categorisation -

Categorisation presented a challenge in the form. The problem is easy enough to state: ask the user for one or more terms under which the organisation falls. These terms are taken from a fixed vocabulary, which ensures that these descriptions are easily comparable (although this requires some effort on the part of the vocabulary designer).

We needed the form to have the following characteristics:

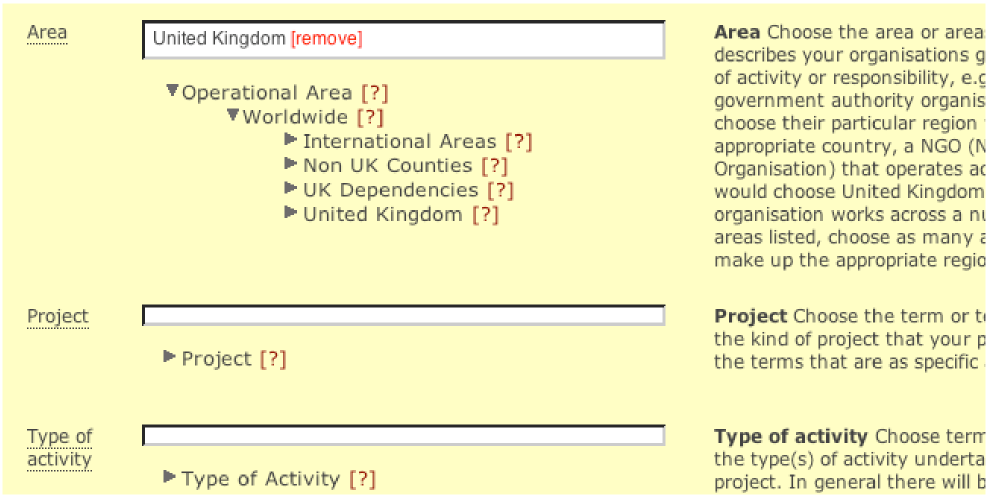
- Users can only pick terms from the vocabulary.
- Ensure that users can find the relevant terms easily.
- Provide a means for users to make an informed choice of terms.

The first restriction precluded the use of simple text field. The second and third requirements were problematic because of the size of the vocabularies, which prevented simple check boxes and descriptions. To illustrate: the operational area vocabulary contains terms for many countries in the world, and regions within the UK. This list would be substantially larger than the rest of the form by itself.

One option would have been to allow the user to search for terms, narrowing down the available options. This level of interaction is, in practice, rather difficult in web forms. What we decided to do instead was present the terms in a tree. This was possible since the vocabularies weren't too large (they can be comfortably downloaded to a web browser) and they also fitted a tree structure.

The result, from the user's perspective, is a process of narrowing down categories, from the more general (e.g. UK) to the specific (Birmingham). Each node in the tree - a term - also provided a link to more detailed information about the term; something which the tree couldn't comfortably accommodate.

The first version of the form is shown below:



Area

United Kingdom [remove]

▼Operational Area [?]
  ▼Worldwide [?]
    ▶ International Areas [?]
    ▶ Non UK Counties [?]
    ▶ UK Dependencies [?]
    ▶ United Kingdom [?]

Area Choose the area or area describes your organisations g of activity or responsibility, e.g government authority organis choose their particular region appropriate country, a NGO (N Organisation) that operates ac would choose United Kingdom organisation works across a nu areas listed, choose as many a make up the appropriate regio

Project

▶ Project [?]

Project Choose the term or t the kind of project that your p the terms that are as specific

Type of activity

▶ Type of Activity [?]

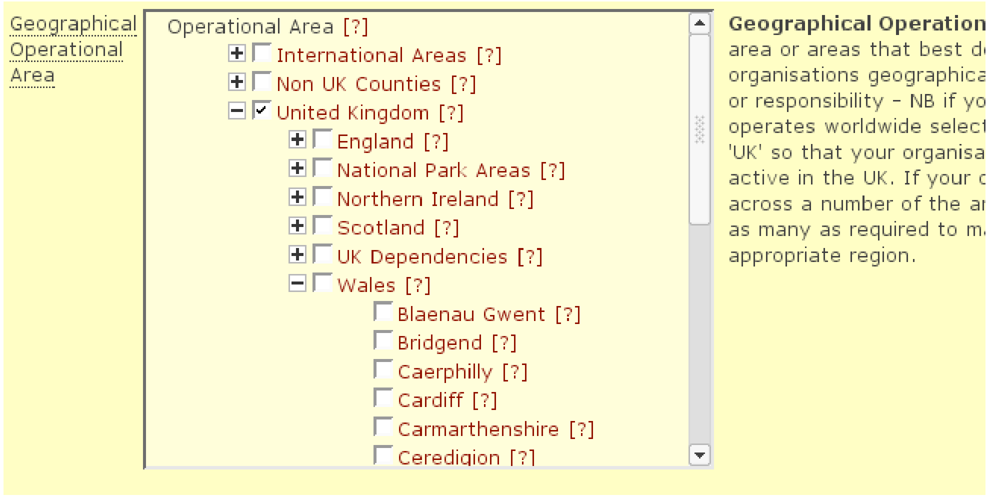
Type of activity Choose term the type(s) of activity underta project. In general there will b

Figure 7 - the original tree control

The initial state can be seen in the Project and Type of Activity fields. When items are selected (as with 'United Kingdom') they are added to a list above. One advantage of this is that if the data is reloaded it is immediately obvious what has been chosen. More detailed information can be shown (in a separate window) by clicking on the `[?]`.

Testing (see ┤ below) revealed that this was pretty confusing. Firstly, as with all custom controls, it wasn't obvious *how* it worked. It wasn't at all clear that the tree roots contained anything. Also as one explored the tree the list could disappear out of the top of the window, so no feedback is given when a term was selected. Indeed the entire form would grow alarmingly.

The final version was changed substantially. The first level of the tree was expanded, so it was immediately clear that there was something to find. The tree was also constrained in a fixed-height box, which would scroll when the tree became too big. This limited the effect of expanding the tree on the rest of the form. Finally, rather than adding items to a separate list each term had a checkbox. This made it immediately obvious that one or more items could be selected.



Geographical Operational Area

Operational Area [?]
  ⊞ ☐ International Areas [?]
  ⊞ ☐ Non UK Counties [?]
  ⊟ ☑ United Kingdom [?]
    ⊞ ☐ England [?]
    ⊞ ☐ National Park Areas [?]
    ⊞ ☐ Northern Ireland [?]
    ⊞ ☐ Scotland [?]
    ⊞ ☐ UK Dependencies [?]
    ⊟ ☐ Wales [?]
      ☐ Blaenau Gwent [?]
      ☐ Bridgend [?]
      ☐ Caerphilly [?]
      ☐ Cardiff [?]
      ☐ Carmarthenshire [?]
      ☐ Ceredigion [?]

Geographical Operation area or areas that best d organisations geographica or responsibility - NB if yo operates worldwide select 'UK' so that your organisa active in the UK. If your c across a number of the ar as many as required to m appropriate region.

Figure 8 - the current tree control

The configuration file links a field to a vocabulary using a `hasVocab` attribute; this points to a bNode containing details of the RDFS class used for tree nodes (`hasConceptType`) and the RDF property used to link child nodes with their parents (`hasBroaderTerm`). For SKOS vocabularies, the concept type was `skos:Concept` and the broader term `skos:broader`; for operational area the corresponding notions were `swed_oa:Area` and `swed_oa:contained_within`; for OWL vocabularies `owl:class` and `rdfs:subClassOf` would be sufficient.

UI for relationships -

Another complex form element was the 'Relations to other ...' field. This field asked the user to give a relation (say 'part of') and a thing (organisation, project) the subject of the form is related to.

Why was this field more difficult than the others? The main problem is that it was a different kind of task: the form is concerned with information intrinsic to the organisation and something the user ought to be well equipped to fill out. Mistakes here are relatively harmless. Relations, by contrast, provide links to other entities. The user may know little about them, and a mistake is costly since the link will fail.

The referential integrity issue was exacerbated by the decentralised nature of the our project. We wanted people to be able to describe their organisations even in the absence of an aggregated store, where they could find the related object.

What we ended up with is this:

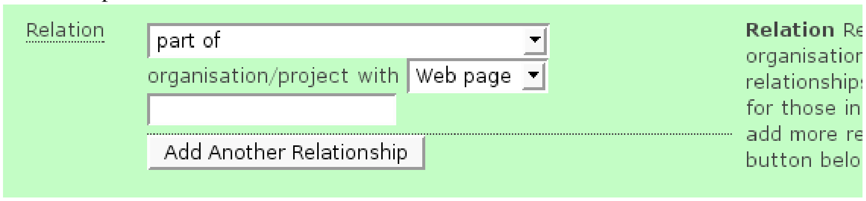

Relation: part of
organisation/project with: Web page

Figure 9 - the relations field

The user was asked to pick a relation type, and give some minimal information to identify the object of the relation. This could be web page (represented as an HTTP URI), an email address (a mailto: URI) or a telephone number (a tel: URI). A JavaScript routine allowed users to dynamically clone the field so that relations could be added (or removed).

In the configuration file, relations were signalled by use of a `indirectProperty` attribute (set to `hasPrimaryURL`), as described earlier. Like classification fields, relationship fields use `hasVocab` to draw in an external schema, but here it is used to populate the drop down box (ie the controlled RDF Property).

UI for preview -

After form submission, the form values are used to construct an RDF model (which is saved locally). At this point, the user is given a chance to re-edit the ┤prorg data before registration. The RDF is rendered in a user friendly manner. At first, *renderers* were used. This enables (for example) address fields to be aggregated and pretty printed, in a similar manner to the portal. However after initial user feedback it seemed that people would prefer a summary of their entered data in the same order and format in which they entered it. Therefore the constructed RDF was displayed according to the preferences encoded in the configuration file.

Form reloading -

If a user wanted to re-edit their data, they were presented with the form again. However, browser 'back' functionality is not guaranteed to save much of the DOM state (important for classification and relationship data) and so a reload functionality was implemented. This allows the form to be displayed with an RDF model and the configuration file specified as request parameters.

User Feedback and Testing -

We conducted a small scale user evaluation study of the data entry processes and forms/e-mails, in partnership with the Natural History Museum in London. We asked five representatives of different museum departments to evaluate the system by using it to enter in formation about their department or projects within their departments. This was very helpful in highlighting a number of both detailed and high level usability issues with the initial prototype of the processes and forms/e-mails. Significant issues included:

1. **Instructions** - the data entry process for SWED are un-usual in a number of respects e.g. the fact that the data file is not stored centrally but by the organisations/projects themselves, the explicit copyright notice, the classification using small (relative to large scale thesauri) but nonetheless significant thesauri and the comprehensiveness of the information relative to the majority of directories. This means that instructions needed to be very carefully written and made as easy and prominent as possible, especially as the users tend not to read instructions unless they feel that they are stuck in some way. Our approach was to place generic instructions above

any data entry elements and specific instructions to the right of the data entry elements - rather than giving the user a link to the instructions. While the latter approach retains screen real estate effectively we felt that user may not then actually read the instructions, which include important points. This approach did seem to work, with users reading the instructions to help decide *exactly* what was required.

2. **Titles of the properties** - in a number of cases the titles of the properties were originally ambiguous e.g. name as opposed to 'project name', and 'operational range' rather than 'geographical operational range' causing some confusion. User testing seems the most effective way to identify such problems.
3. **Selection of thesauri terms** - The second design for the selection of thesauri terms (following the move from the nanotree based system used on the portal because of issues with loading times) was a simple tree (see ┤ Fig 7 above) that started in a collapsed state (i.e. only the root term showing) and where the user clicked on and icon to right to open a branch and the term itself to select it, the selected terms would appear in a text box above the tree. This caused significant confusion, e.g. it was not realised that the tree could open or that clicking on the terms was the way to select them (even though the instructions noted this). It also meant that the form became very long and users could not see the text box when clicking on terms low down in the tree. This was amended to its current state (see ┤ Fig 8) changing the design significantly with the tree in a vertically scrollable box, with check boxes for selection. This seems to overcome the problems with the initial design. However for large or/and complex thesauri this interface may be inadequate and a we felt a specific thesaurus browser, perhaps in a popup window, or a multi-page form, might be more appropriate.
4. **Selection of the thesauri terms** - from the thesauri was often time consuming, especially in a few cases the number of relevant terms within a classification scheme was very high, (e.g. the 'topics of interest' property for the library department).
5. **Collecting data** - It seemed likely that in some cases it would be necessary for users to collect information before being able to complete the form - and a mechanism for them to return to edit the form would be useful. As the system stands this is possible as the reload system could fulfill this function and this could be implemented as a future enhancement.
6. **Auto-completion** - An issue raised by four of the five participants was that of pre-loading data where it had already been entered into the system e.g. where a department was adding its data and the main organisation already has an entry. It was felt to be extremely frustrating in these cases (at the Natural History Museum departments might have 10 or more projects). This is clearly possible using an adapted version of the reload tool and an interface to locate the source record to use to pre-populate relevant fields e.g. contact details. This could be implemented as a future enhancement.
7. **The post 'submit' screen** - after the user submits the date entry form data, they are taken to a page to check their data they have entered and instructions for the next stage. As this process is unusual some users found this complex and made suggestion for simplifying the text. The unusualness of the process was certainly an issue and seems to necessitate detailed and clear instructions, which must be balanced over making the page look so full of text as to be daunting.
8. **E-mail response and internal validation** - the e-mail response to the final submission was the most unusual aspect of the process for most users, and once again the text of this message needed to be made very simple and clear. It was also noted that any information would have to go through the museum's normal QA systems for approval before it could be published.
9. **Uploading and registering the location of the RDF file** - this part of the process was not covered in this user evaluation.

Lessons learnt - data entry -

Workflow emphasises distributed nature of portal

The users of the data creation functionality were emailed (a link to) their RDF for hosting on their own sites. Once hosted, they (or more likely their webmasters) returned to the data creation site to register their RDF. This workflow underlines the distributed nature of the portal; that is, organisations truly own their data. In addition the data creation functionality is logically distinct from the portal. It is a separate web application, with its own look and feel. This again shows its distinctiveness from traditional portals.

The lessons learnt from semantic blogging were useful

In the semantic blogging demonstrator, we developed a notion of metadata dependent views (semantic views), controlled by a configuration file. This notion was developed and extended here. In addition, the data model (blog entries are annotations over information items) was a useful starting point, although here of course there are only information items, not blog entries.

The balance of general versus tailored configurations is an open challenge

The configuration files enabled us to have a single point of entry into the data creation functionality, rather than, say, separate JSPs for the different prorg types. In principle this could be extended to data entry of completely different information resources. All that is needed is for a notion of fields and field groups. Even the stylesheet is specified by the schema.

However, our experience with classifications and relationships showed that some forms of data entry require post hoc extensions to the configuration schema in order to maintain the desired level of functionality. It could be argued that the same result could have been achieved with dedicated JSPs more easily but at the cost of reduced flexibility. In addition, the cost of maintaining 3 separate JSPs was replaced by the cost of maintaining 3 separate configuration files. In fact, the configuration files were easier to maintain, the rendering functionality was more easily tested, and the reload functionality more tractable with the solution we chose.

Nonetheless it will not always be clear where to draw the line between simple customised solutions and complex general ones. Indeed, the complexity of the form is underlined by the fact that we found it necessary to turn server side compression on to get adequate performance on some client platforms.

The freedom of RDF poses a challenge for the constraints of a web form

Reloading provided another valuable lesson from the form. This requires the form to be repopulated with (supplied) RDF given a configuration file. Hence, each form field needs to be associated with some value from the RDF. Some fields has a simple bijective value to field mapping; they each control different properties. But for some fields one needs additional information: for example the telephone and fax numbers only differ in type, everything else about them is the same. We also had difficulties were found in repopulating classification data, relationship data, repeated fields and indirect properties.

One difficulty is that identifying a particular node in RDF, other than by its label or value, is rather hard. XML has a root, ordering, and containment which is more than enough structure to identify nodes (eg by paths from the root). RDF has none of these, so structure must be provided. Reediting is a sufficiently important usability feature that the additional effort in modelling was necessary.
In general, this is the problem of reverse engineering input that gave rise to some specified RDF. In our case the disambiguation clues provided in the configuration file (eg `hasIndirectContext`) were sufficient to reconstruct the form, in particular from RDF that had just been created from that form. It is not clear how successful the reload functionality would be in the case of truly arbitrary input RDF.

Order is important

RDF is by its nature unordered. This brings with it great flexibility but also creates problems where we need to present elements in a certain order. We adopted two solutions. For relationships (where the various options - 'partOf', 'hasPart' etc - should be paired and ordered) the solution chosen was to encode the RDF as a sequence. For classification metadata this was inappropriate since the schema chosen was an external vocabulary on which we could not impose a sequence. We adopted an external utility which ordered the resources by (RDFS) label. The utility was also used to order the fields on the form, except that here we had the option of specifying a priority, thus explicitly ordering them within the configuration schema. Ordering is also important for specialised renderers (eg for address) which were implemented as helper classes.

Validation is important

In an early release, a user inadvertently tested the functionality by entering a URL without an "http://" prefix. It quickly became clear that we needed validation. In some cases (eg email addresses, primary name) this was done at the form level, refusing to submit the form until an acceptable value was entered. In other cases (eg URLs) we implemented a validator to silently fix various common abbreviations (eg adding a tel: prefix to a telephone URI). The validator class name was associated with the field in the configuration form.

We have implemented minimal validation but could do much more. It is probably best to assume that if users will get it 'wrong' and this should be handled gracefully. RDF models can be unforgiving of (eg) malformed URIs.

Identifying organisations is an open issue

It is not always clear what URI to assign to a prorg. We used a concatenation of the prorg primary name, appended to a canonical SWED namespace. This solution worked well for the portal since all prorgs had names (this was enforced by the data creation form) and in edge cases where the name consisted of no suitable characters (eg all digits) a URN was constructed. However, tying prorgs together (eg by relationships) was more problematic. We adopted a solution of using primary URL as an inverse functional property (so two prorgs with the same primary URL were assumed to be the same).

The notion of primary URL was extended to cover email addresses and telephone numbers (for prorgs without a web presence). However, this requires the creator of a relationship to know the primary URL of the destination prorg. We discussed (but did not implement) various heuristics to tie together organisations with (for example) matching names but mismatching primary URLs. This is one area where a centralised (or distributed) resolver service might be used to assist the user, or where post hoc correction procedures might be useful.

Classifications are hard to get right

Fields using classification metadata (drawn from a vocabulary) were relatively complex to configure. More importantly however, they are non-intuitive for the user. We undertook a significant modification to the UI after user testing; this will hopefully improve users' understanding.

It remains to be seen how successful the tree field is. It certainly doesn't scale to large vocabularies. It requires some hunting by the user, and a suitable vocabulary: a collection of unrelated concepts won't work. However it does work for our vocabularies, which don't appear atypical for categorisation.

Relationships are hard to get right

The relationship UI was about as simple as we could make it. But it was not not well received by our initial users. In particular, it was not clear that the form had moved on, that the description task had finished, and the user was now engaged in relating this thing to other bodies. Additionally the question seemed a little odd (why give my relation to a web page of the project?) unless you understood how the model worked.

Perhaps this task should have been given its own page, where more space could have been devoted to explaining how to describe relationships.

User feedback

In general users were positive about the data entry functionality though they certainly needed clear instructions. The data creation process is rather different from the semantic portal as it is by nature 'one off' (users expect to fill out the form only once). Therefore we had some leeway to make the form longer, and to provide clear, rather than lightweight, instructions.

The form functionality is admittedly complex. The non-intuitive notions of decentralization, classification and relationships are embedded in the form and need to be appreciated by the user to access the functionality effectively. However, there is room for optimism. These are also key semantic web notions - if users understand them, and if they see value in this way of doing things, then they can be said, at some level, to have gained an appreciation of the semantic web. And that is a key goal of the demonstrator.

5 Deployment and dissemination

5.1 Status

The complete SWED demonstrator, built using the above software components, has been deployed as an externally accessible web site at http://www.swed.org.uk. This is hosted on a server at HP Laboratories as a demonstration service. The software developed for the demonstrator is available with documentation and full source code from the *Technical Resources* section of that site.

Some development work on the software will continue until the end of the SWAD-E project. As well as small modifications in response to user feedback we would also like to extend the portal functionality so that the aggregated data is re-exported in a way that other services could use (such as via a Joseki [JOSEKI] server).

In the next few sections we will describe the results of preliminary user evaluations, the impact the demonstrator has had in assisting Semantic Web dissemination and the future direction for the demonstrator.

5.2 User evaluation

We have conducted a small number of preliminary user evaluation sessions of the portal and have received some solicited and unsolicited feedback.

The results of the user testing and the feedback provided details of specific usability points about the initial interface for the prototype (the vast majority of which have since had solutions implemented) and a few points about functionality.

Overall the the faceted browse approach to exploring the information seems intuitive to all of the users who have evaluated the system.

There are some specific issues related to browse/search approach that were of note:

1. It is not possible to create conjunctive queries on terms from the same facet. This is also the case with other existing faceted browse systems that we are aware of including both complex systems such as the Flamenco browser [FLAMENCO] and more simple systems such as the VOSCUR directory of Bristol's community & voluntary groups and social enterprises [VOSCUR]. We will explore potential solutions to this problem over the coming months.
2. The choice of terms and structure of the thesauri are significant to users. In particular it can be difficult for users to locate specific terms related to specific queries that they may have, this is especially the case for larger and conceptually rich vocabularies such as 'topic of interest'. Our attempt to overcome this by using multi-hierarchies (concepts can appear in more than one place in the tree) seem to be effective, however more focused user testing would be required to verify this.
3. The title of the search results page was noted by one user to be the same for all queries and therefore they could not differentiate between them in the browser back button dropdown list. How to solve this requires some thought and experimentation since it is not immediately clear how to construct titles that would solve the problem as the queries (build from the user browsing the facets) can be complex.

We plan to conduct more user evaluation sessions over the next months. This will enable us to make more comprehensive review of usability issues which can feed into the ongoing development of the system.

In additional to the direct user feed there there two other areas of the user interface design that could warrant further investigation in the future. Firstly, alternative user interface paradigms for exploring the thesaurus hierarchies might be useful, the current expandable tree approach is adequate but not ideal. Secondly, our solutions to the presentation of provenance information to the user just a first approximation to a full solution. This is a general issue for semantic web applications that present information to a user that has been aggregated from multiple heterogeneous sources. Both of these issues were noted earlier in the report and we just repeat them here for completeness.

5.3 Moving forward

The SWED system has been based on a real world problem and expressed need, as identified by our preliminary survey of biodiversity/wildlife information in the UK [SHAB]. We have worked from the beginning of the project with the Environment Council [ENV COUNCIL] in the UK and more recently with the Natural History Museum in London [NHM] and VOSCUR [VOSCUR] based in Bristol. From the conception of the project we have aimed to ensure that the SWED approach and system are taken forward once the SWAD-E project comes to an end.

The involvement of the Environment Council as a key network organisation in the UK and an organisation that had a background in both the production of environmental directories and a need for a SWED type directory has been very important to that goal. The involvement of Natural History Museum too has been very valuable; the museum has very extensive experience in data integration of natural history data including information derived from multiple organisations and projects. Their support and help in defining a specific example of a potential extension of the SWED core system to include information about museum collections has been particularly valuable.

Involvement with VOSCUR (a voluntary sector community network organisation) has helped understanding the issues related to more locally based and smaller organisations.

At present (late August 2004) there are three parallel routes being explored that may take the environmentally focused applications of the project forward.

1. Generic ongoing development : It is planned that a group of lead organisations (coordinated by the Environment Council) will conduct that basic work with minimal initial funding and then perhaps make recommendations/a funding bid, to implement any more involved generic improvements to the system or additional tool kits (e.g. to help migrate legacy content).
2. Short term maintenance: In the short term HP Labs will continue hosting the demonstration

SWED system. However we are meeting with the Environment Council to discuss the how best to move to a medium term hosting solution that would probably also be part of the ongoing development from (1) above.
3. Specific Implementations: At present we have interest from both the Environment Council and the Natural History Museum in the development of directory type services based on the SWED architecture. Details of these should be forthcoming over the next months.

The involvement of external organisations have been very useful in helping keep the project focused on meeting the needs of real organisations and in genuine contexts, as well as providing an avenue for the ongoing development of the outputs of the system.

More generally, the semantic portals approach and the software tools developed have been to designed to be very general and we expected to see them reused in other applications. We note in the next section that we already have interest (both from within HP and from external partners and customers) in using the tools and approach in other settings.

5.4 Impact

The demonstrator has been very successful in meeting its prime purpose of communicating the nature and value of the semantic web. Some specific instances of this are:

Natural History Museum

The Natural History Museum is a hub for data exchange relating to the natural world and is involved in several projects involving integration of data from multiple sources such as BioCase [-| BIOCASE]. They had been previously aware of RDF and Semantic Web but were concerned that these technologies were too complex and impractical for real world world. As a result of the SWED demonstrator they can see that practical semantic web applications are deployable now. Furthermore, the benefits of the decentralized approach that the semantic web enables is very appealing in their domain. The ability for each provider to host their own data and yet have the data be cross-linked, and one provider be able to annotate and enrich another provider's data, all seem relevant.

As a direct result of this demonstrator they have asked us to participate in a group seeking to establish a similar infrastructure for the sharing and integration of collection-level descriptions, initially within the UK.

Environment council

As noted above the Environment Council have been very supportive of the project from its inception and would like to explore application of the same approach to other related areas including organizations across Europe.

JISC

As a result of the demonstrator we were able to offer input to a JISC (Joint Information Systems Committee: [-| JISC]) sponsored study of information portals. Again the decentralized nature of the semantic web approach was seen as an interesting aspect of the approach.

Research Project Proposals

As a result of the demonstrator a number of funding proposals under e-science/e-research programmes have been made that describe systems that would use the basic architecture and software from the semantic portals project. These include bids related to reserach 'administration' information integration (e.g. about projects, departments, publications, researchers), research results and information co-ordination and integration and conference information portal systems.

HP External

HP has a well established and very active semantic web research group. Since the deployment of the demonstrator we have been able to use it as one of our tools for explaining and illustrating the nature of the semantic web to customers and partners of HP to good effect.

In particular we have worked with HP's consulting arm to successfully bid for a commercial project in educational metadata management on the basis of the demonstrator.

HP Internal

As part of our work in disseminating awareness and knowledge of the semantic web within HP we developed a demonstration knowledge management application that combined the semantic blogging and semantic portals demonstrator. This confirmed that the web portal software is flexible and configurable. The resulting internal demonstrator has also been a very effective communication tool and as a result we expect to begin further pilot projects in the broad area of knowledge management.

6 Conclusions

The demonstrator project has been successful at several levels.

In terms of the SWED environmental directory application itself we have gained good feedback from initial users and there is substantial interest in taking the work forward from a demonstrator to a supported service.

In terms of the overall SWAD-E objectives it has proved a useful tool in demonstrating the nature of the semantic web. Several features of the demonstrator help with this. First it is a relatively simple application which is quick to show. Secondly the fact that the data is integrated from multiple sources helps to communicate the decentralized nature of the semantic web and the value and meaning of semantic integration. Finally its use of a mix of ontologies and thesauri help to illustrate that aspect of the semantic web in a way that is quite visible within the application.

As a result of successful use of the demonstrator there are several activities planned which either extend the ideas into neighbouring domains (such as the collection level description of natural history resources, or the integration of educational metadata) or more into general applications (such as knowledge management).

Finally in terms of software components both the data collection and web display tools developed for this demonstrator are reusable, customizable components which will hopefully prove a useful resource for the community.

A References

[AKTIVESPACE]
AKTiveSpace Ontology
http://www.aktors.org/publications/ontology/

[ANALYSIS]
12.1.1 Open demonstrators: Semantic web applications - analysis and selection
Dave Reynolds , Steve Cayzer, Ian Dickinson, Paul Shabajee
http://www.w3.org/2001/sw/Europe/reports/chosen_demos_rationale_report/hp-applications-selectic

[BIOCASE]
BioCase A Biological Collection Access Service for Europe
http://www.biocase.org/

[BIOTHES]
Biocomplexity Thesaurus
http://thesaurus.nbii.gov/

[CC]
Charity Commission
http://www.charity-commission.gov.uk/registeredcharities/first.asp

[ENV COUNCIL]
UK Environment Council
http://www.the-environment-council.org.uk/

[ENVIROLINK]
Envirolink UK, East of England Ecodirectory
http://www.ecodirectory.org/directorymaster/t_index.php

[ETT]
Environmental Thesaurus and Terminology Workshop, Geneva
http://esw.w3.org/mt/esw/archives/000052.html

[FLAMENCO]
Hierarchical Faceted Metadata in Site Search Interfaces
Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ping Yee, in the CHI

2002
See also: http://bailando.sims.berkeley.edu/flamenco.html

[FOAF]
The Friend of a Friend (FOAF) project.
http://www.foaf-project.org/

[GEMET]
GEneral Multilingual Environmental Thesaurus
http://www.eionet.eu.int/GEMET

[GETTY]
Getty Thesaurus of Geographic Names
http://www.getty.edu/research/conducting_research/vocabularies/tgn/

[GIG]
UK GIGateway
http://www.gigateway.org.uk/

[ISIC]
International Standard Industrial Classification
http://unstats.un.org/unsd/cr/registry/regct.asp?Lg=1

[JENA]
Jena – A Semantic Web Framework for Java
http://jena.sourceforge.net/

[JISC]
The Joint Information Systems Committee
http://www.jisc.ac.uk/

[JOSEKI]
Joseki: Jena RDF web server
http://www.joseki.org/

[JSP]
J2EE JavaServer Pages Technology
http://java.sun.com/products/jsp/

[NHM]
Natural History Museum
http://www.nhm.ac.uk/

[OWL]
OWL Web Ontology Language Reference
W3C Recommendation 10 February 2004
Editors: Mike Dean, Guus Schreiber
http://www.w3.org/TR/owl-ref/

[PORTAL DOCUMENTATION]
SWAD-E Portal Structure, Dave Reynolds
http://www.swed.org.uk/swed/doc/portal-structure.html

[PORTAL RESOURCES]
SWAD-E Portal Resources, Dave Reynolds
http://www.swed.org.uk/swed/swed_technical_resources.htm

[PROTEGE-OWL]
The Protégé OWL plugin
http://protege.stanford.edu/plugins/owl/index.html

[RDF]
The Resource Description Framework
http://www.w3.org/RDF/
http://www.w3.org/2001/SW/RDFCore/

[RDFS]
RDF Vocabulary Description Language 1.0: RDF Schema
http://www.w3.org/TR/rdf-schema/

[REDLAND]
REDLAND RDF Application Framework, Dave Beckett
http://www.redland.opensource.ac.uk/

[REQUIREMENTS]
12.1.5 Semantic Portals - Requirements Specification
Dave Reynolds, Paul Shabajee
http://www.w3.org/2001/sw/Europe/reports/requirements-demo-2/

[SEAL]
Semantic portal - The SEAL approach,
Maedche, A., Staab, S., Stojanoic, N., Studer, R., Sure, Y.,
in *Creating the Semantic Web* Fensel, D et al (eds.) MIT Press, MA, Cambridge, 2001.

[SHAB]
Shabajee, P. (2004) *Summary Report from SWARA Survey of Biodiversity/Wildlife Information in the UK*, HP (Hewlett Packard) Labs Technical Report, HPL-2004-58.
Available: http://www.hpl.hp.com/techreports/2004/HPL-2004-58.html

[SKOS]
Simple Knowledge Organization Schema, SWAD-E Thesaurus activity
http://www.w3.org/2001/sw/Europe/reports/thes/

[TOMCAT]
The Apache Jakarta Tomcat project
http://jakarta.apache.org/tomcat/

[VCARD]
Representing vCard Objects in RDF/XML
Renato Iannella, W3C Note 22 February 2001
http://www.w3.org/TR/vcard-rdf

[VCARD-DISCUSSION]
Discussion on applicability of vCard to organizations on CETIS-METADATA jisc mail list for July 2004
http://www.jiscmail.ac.uk/cgi-bin/webadmin?A2=ind0407&L=cetis-metadata&T=0&P=1396

[VELOCITY]
Apache Jakara Velocity
http://jakarta.apache.org/velocity/

[VOSCUR]
Voscur, Council of Voluntary Services (CVS) for Bristol
Home: http://www.voscur.org/
Directory: http://www.voscur.org/directory/VDir/VDirHome.html

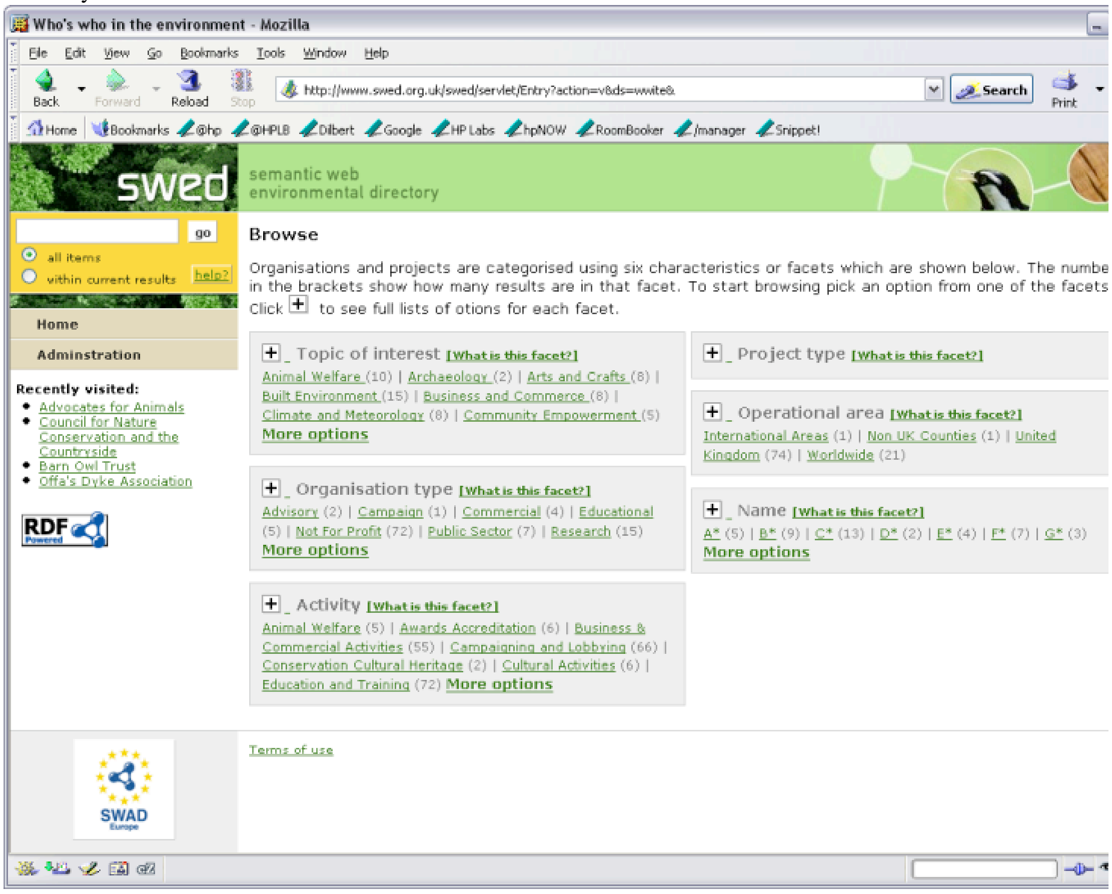
[WEB-PORTALS]
Querying Community Web Portals (2000)
Kavournarakis, G., Christophides, V., Plexousakis, D., and Alexaki, S.,
http://citeseer.nj.nec.com/karvounarakis00querying.html

[WWITE]
Who's Who in the Environment,
PC Disk version 1.0, © Environment Council, 1998

B Demonstrator screen shots

The current demonstrator may be visited at: http://www.swed.org.uk.

The initial pages for the demonstrator provide information on the project, the technology and links to technical resources such as downloadable versions of the software. The first page of the directory itself looks like:


Who's who in the environment - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.swed.org.uk/swed/servlet/Entry?action=v&ds=write&
Search
Print
Home Bookmarks @hp @HPLB Dilbert Google HP Labs hpNOW RoomBooker /manager Snippet!
swed
semantic web environmental directory
go
all items
within current results
help?
Home
Administration
RDF Powered
SWAD Europe

Browse

Organisations and projects are categorised using six characteristics or facets which are shown below. The numbe in the brackets show how many results are in that facet. To start browsing pick an option from one of the facets Click + to see full lists of otions for each facet.

Recently visited:
- Advocates for Animals
- Council for Nature Conservation and the Countryside
- Barn Owl Trust
- Offa's Dyke Association

+ _ Topic of interest [What is this facet?]
Animal Welfare (10) | Archaeology (2) | Arts and Crafts (8) | Built Environment (15) | Business and Commerce (8) | Climate and Meteorology (8) | Community Empowerment (5)
More options

+ _ Organisation type [What is this facet?]
Advisory (2) | Campaign (1) | Commercial (4) | Educational (5) | Not For Profit (72) | Public Sector (7) | Research (15)
More options

+ _ Activity [What is this facet?]
Animal Welfare (5) | Awards Accreditation (6) | Business & Commercial Activities (55) | Campaigning and Lobbying (66) | Conservation Cultural Heritage (2) | Cultural Activities (6) | Education and Training (72) More options

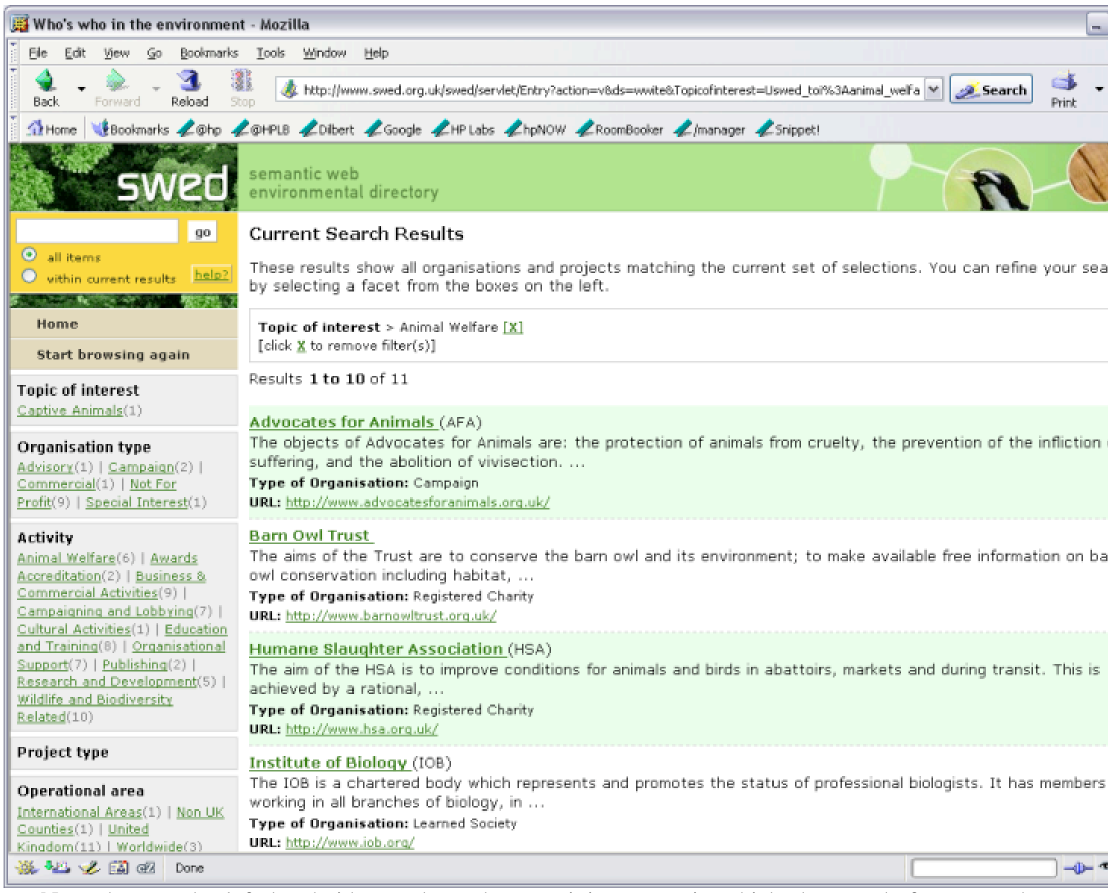
+ _ Project type [What is this facet?]

+ _ Operational area [What is this facet?]
International Areas (1) | Non UK Counties (1) | United Kingdom (74) | Worldwide (21)

+ _ Name [What is this facet?]
A* (5) | B* (9) | C* (13) | D* (2) | E* (4) | F* (7) | G* (3)
More options

Terms of use

This shows six different "facets" around which the information is organized. Selecting an item in a facet such as the entry *Animal Welfare* in the *Topics of interest* facet shows a list of matching organizations:


Who's who in the environment - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.swed.org.uk/swed/servlet/Entry?action=v&ds=wwite&Topicofinterest=Uswed_toi%3Aanimal_welfa
Search Print
Home Bookmarks @hp @HPLB Dilbert Google HP Labs hpNOW RoomBooker /manager Snippet!
swed
semantic web
environmental directory
go
all items
within current results
help?
Done

Home

Start browsing again

Topic of interest
Captive Animals(1)

Organisation type
Advisory(1) | Campaign(2) | Commercial(1) | Not For Profit(9) | Special Interest(1)

Activity
Animal Welfare(6) | Awards Accreditation(2) | Business & Commercial Activities(9) | Campaigning and Lobbying(7) | Cultural Activities(1) | Education and Training(8) | Organisational Support(7) | Publishing(2) | Research and Development(5) | Wildlife and Biodiversity Related(10)

Project type

Operational area
International Areas(1) | Non UK Counties(1) | United Kingdom(11) | Worldwide(3)

Current Search Results

These results show all organisations and projects matching the current set of selections. You can refine your sea
by selecting a facet from the boxes on the left.

Topic of interest > Animal Welfare [X]
[click X to remove filter(s)]

Results **1 to 10** of 11

Advocates for Animals (AFA)
The objects of Advocates for Animals are: the protection of animals from cruelty, the prevention of the infliction
suffering, and the abolition of vivisection. ...
Type of Organisation: Campaign
URL: http://www.advocatesforanimals.org.uk/

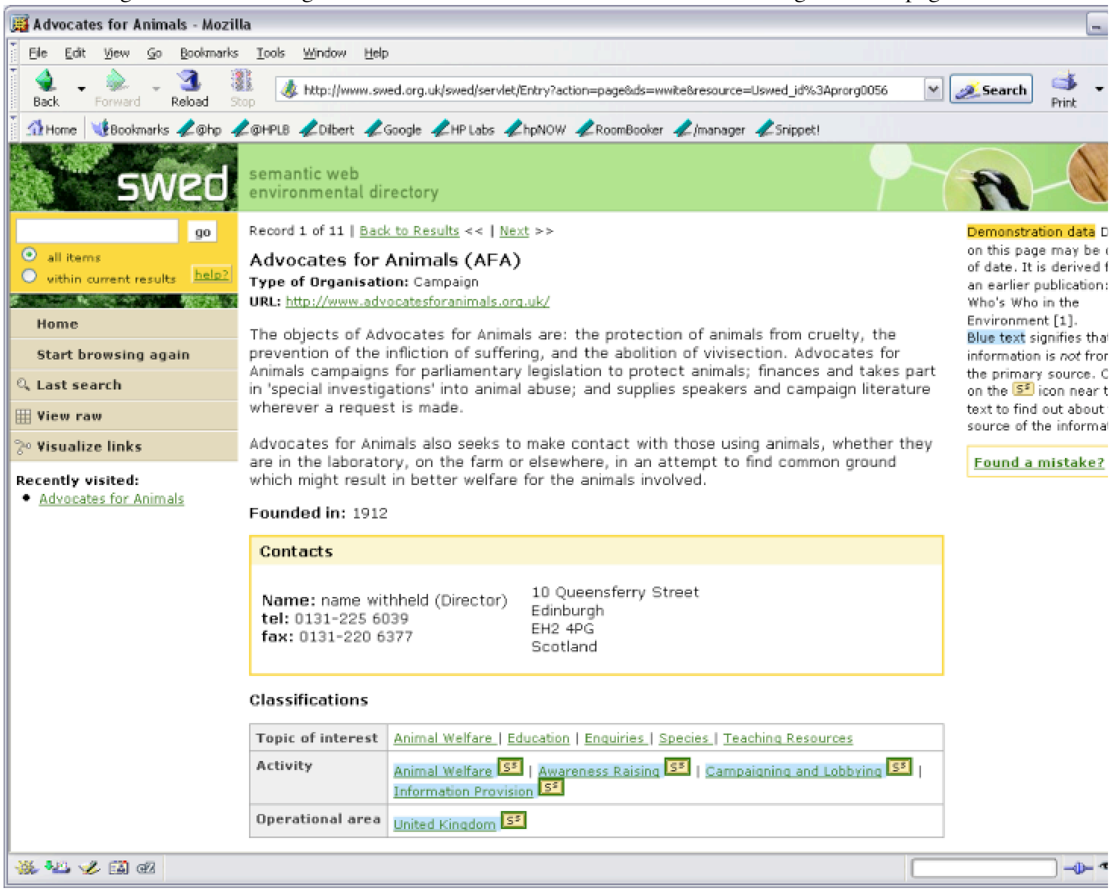
Barn Owl Trust
The aims of the Trust are to conserve the barn owl and its environment; to make available free information on ba
owl conservation including habitat, ...
Type of Organisation: Registered Charity
URL: http://www.barnowltrust.org.uk/

Humane Slaughter Association (HSA)
The aim of the HSA is to improve conditions for animals and birds in abattoirs, markets and during transit. This is
achieved by a rational, ...
Type of Organisation: Registered Charity
URL: http://www.hsa.org.uk/

Institute of Biology (IOB)
The IOB is a chartered body which represents and promotes the status of professional biologists. It has members
working in all branches of biology, in ...
Type of Organisation: Learned Society
URL: http://www.iob.org/

Note that on the left hand side we show the remaining ways in which the search facets can be refined.

Selecting an individual organization from the list of matches shows the organization page:


Advocates for Animals - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://www.swed.org.uk/swed/servlet/Entry?action=page&ds=wwite&resource=Uswed_id%3Aprorg0056
Search Print
Home Bookmarks @hp @HPLB Dilbert Google HP Labs hpNOW RoomBooker /manager Snippet!
swed
semantic web
environmental directory
go
all items
within current results
help?

Home

Start browsing again

Last search

View raw

Visualize links

Recently visited:
- Advocates for Animals

Record 1 of 11 | Back to Results << | Next >>

Advocates for Animals (AFA)
Type of Organisation: Campaign
URL: http://www.advocatesforanimals.org.uk/

The objects of Advocates for Animals are: the protection of animals from cruelty, the prevention of the infliction of suffering, and the abolition of vivisection. Advocates for Animals campaigns for parliamentary legislation to protect animals; finances and takes part in 'special investigations' into animal abuse; and supplies speakers and campaign literature wherever a request is made.

Advocates for Animals also seeks to make contact with those using animals, whether they are in the laboratory, on the farm or elsewhere, in an attempt to find common ground which might result in better welfare for the animals involved.

Founded in: 1912

Contacts

Name: name withheld (Director)
tel: 0131-225 6039
fax: 0131-220 6377

10 Queensferry Street
Edinburgh
EH2 4PG
Scotland

Classifications

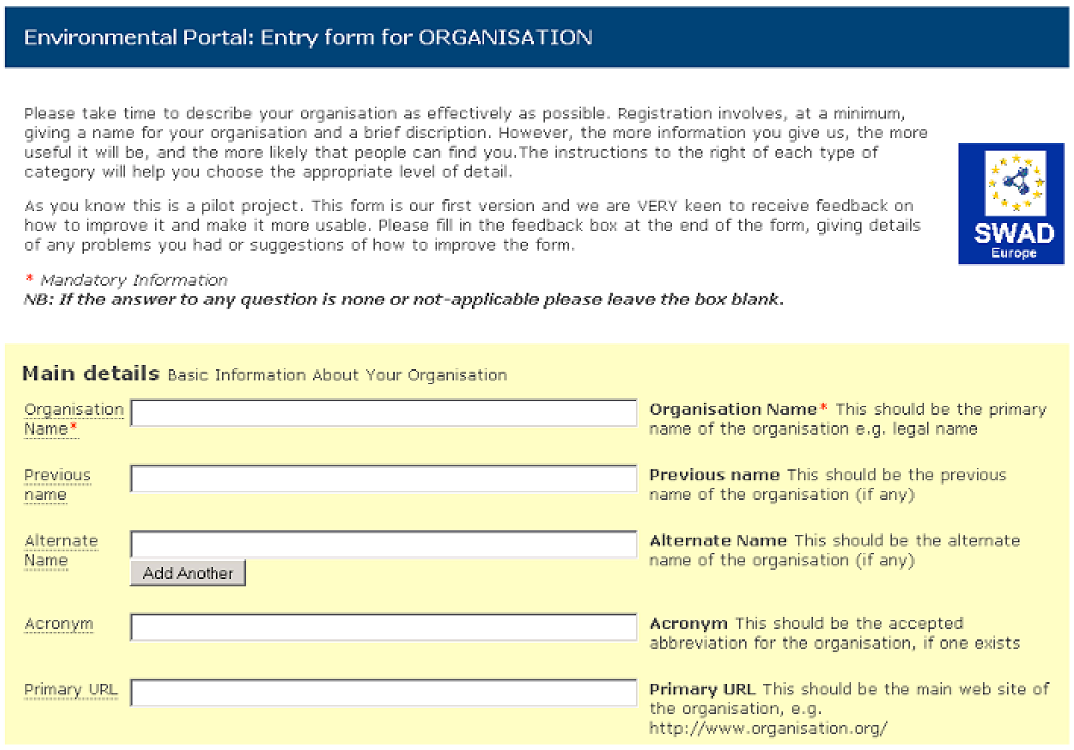
Topic of interest	Animal Welfare | Education | Enquiries | Species | Teaching Resources
Activity	Animal Welfare S² | Awareness Raising S² | Campaigning and Lobbying S² | Information Provision S²
Operational area	United Kingdom S²

Demonstration data D
on this page may be
of date. It is derived f
an earlier publication:
Who's Who in the
Environment [1].
Blue text signifies tha
information is *not* fro
the primary source. C
on the S² icon near t
text to find out about
source of the informa

Found a mistake?

This information, like all the rest, is stored in RDF and converted to a readable web page by the browser software. Note that the provenance of the different parts of the data is highlighted, indicating that this information on this page is a combination of information from different sources.

The initial (start of the) data creation form for organisations looks like this:



Environmental Portal: Entry form for ORGANISATION

Please take time to describe your organisation as effectively as possible. Registration involves, at a minimum, giving a name for your organisation and a brief discription. However, the more information you give us, the more useful it will be, and the more likely that people can find you.The instructions to the right of each type of category will help you choose the appropriate level of detail.

As you know this is a pilot project. This form is our first version and we are VERY keen to receive feedback on how to improve it and make it more usable. Please fill in the feedback box at the end of the form, giving details of any problems you had or suggestions of how to improve the form.

* *Mandatory Information*
NB: If the answer to any question is none or not-applicable please leave the box blank.

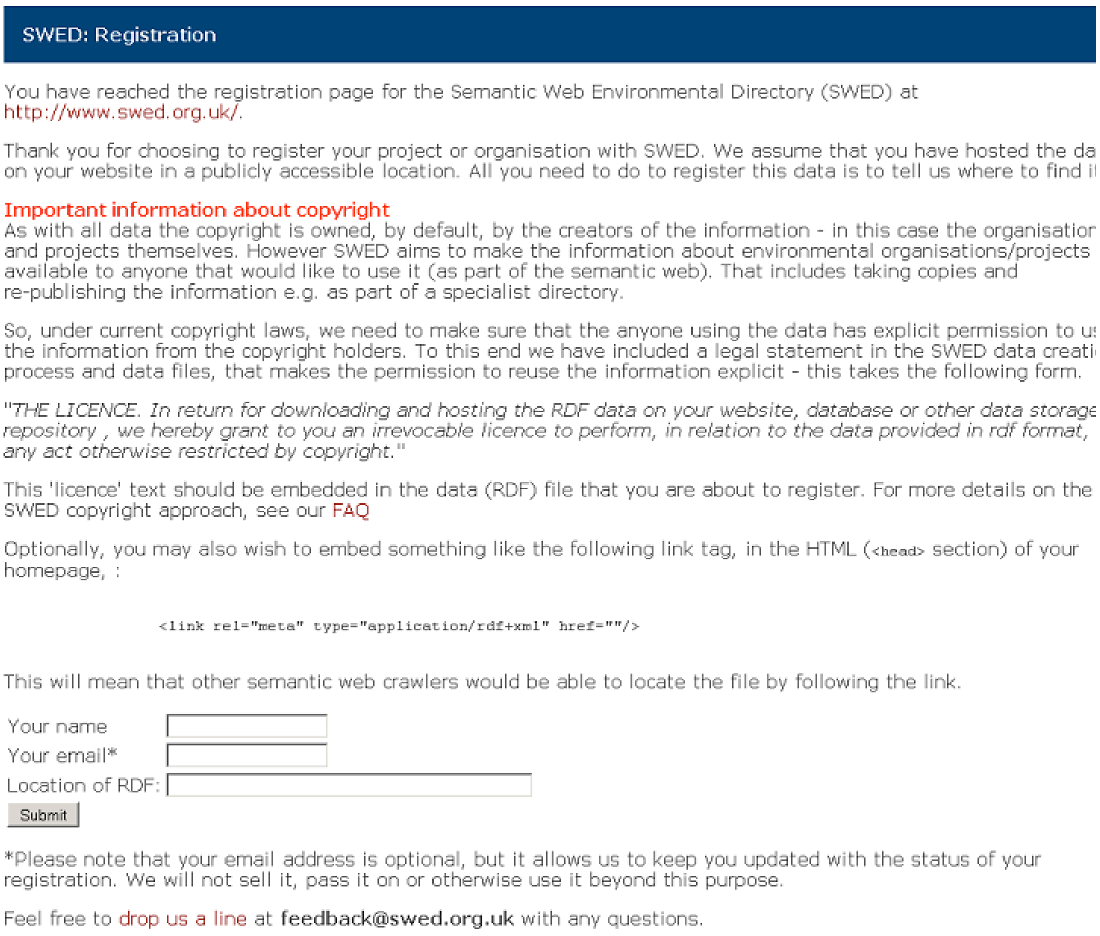
Main details Basic Information About Your Organisation

Organisation Name*		**Organisation Name*** This should be the primary name of the organisation e.g. legal name
Previous name		**Previous name** This should be the previous name of the organisation (if any)
Alternate Name	Add Another	**Alternate Name** This should be the alternate name of the organisation (if any)
Acronym		**Acronym** This should be the accepted abbreviation for the organisation, if one exists
Primary URL		**Primary URL** This should be the main web site of the organisation, e.g. http://www.organisation.org/

When a user fills out this form, they receive a mail similar to the following:

```
This is an automated message from the Semantic Web Environmental Directory (http://www.swed.org.uk/).
Thank you for creating data on your environmental organisation or project.
To complete the registration process, you'll just need to host (upload) this data on your web site
- and tell us about where it is.
You may wish to forward this email to your Webmaster,
or, if you have access to your external webserver, you may wish to do it yourself.
Here's how.
STEP 1: Download the data from here (if your email client supports it, try right clicking and 'download as'):
http://www.swed.org.uk:80/swed_data_entry/rdfit/MYORG.rdf
STEP 2: Host it on your site.
You can put it in any externally accessible location on your server, eg in the same directory as your homepage.
Please remember to make it externally visible (eg set read permissions).
Optionally, you might like to make your web server serve *.rdf files as mime type application/rdf+xml.
(If you don't know how to set mime types don't worry, we'll still be able to harvest your file).
STEP 3: Register the data.
This is a 10 second process which involves giving us the URL of the file.
http://www.swed.org.uk/swed_data_entry/RDFIt?registerit&jsp=register_it.jsp
&uri=http%3A%2F%2Fjena.hpl.hp.com%2F2004%2F02%2Fswed%2Fprorg%23MYORG
```

The data registration form is a very simple form:



SWED: Registration

You have reached the registration page for the Semantic Web Environmental Directory (SWED) at http://www.swed.org.uk/.

Thank you for choosing to register your project or organisation with SWED. We assume that you have hosted the da on your website in a publicly accessible location. All you need to do to register this data is to tell us where to find i

Important information about copyright
As with all data the copyright is owned, by default, by the creators of the information - in this case the organisatior and projects themselves. However SWED aims to make the information about environmental organisations/projects available to anyone that would like to use it (as part of the semantic web). That includes taking copies and re-publishing the information e.g. as part of a specialist directory.

So, under current copyright laws, we need to make sure that the anyone using the data has explicit permission to us the information from the copyright holders. To this end we have included a legal statement in the SWED data creati process and data files, that makes the permission to reuse the information explicit - this takes the following form.

"THE LICENCE. In return for downloading and hosting the RDF data on your website, database or other data storage repository , we hereby grant to you an irrevocable licence to perform, in relation to the data provided in rdf format, any act otherwise restricted by copyright."

This 'licence' text should be embedded in the data (RDF) file that you are about to register. For more details on the SWED copyright approach, see our FAQ

Optionally, you may also wish to embed something like the following link tag, in the HTML (`<head>` section) of your homepage, :

```
<link rel="meta" type="application/rdf+xml" href=""/>
```

This will mean that other semantic web crawlers would be able to locate the file by following the link.

Your name	
Your email*	
Location of RDF:	

Submit

*Please note that your email address is optional, but it allows us to keep you updated with the status of your registration. We will not sell it, pass it on or otherwise use it beyond this purpose.

Feel free to drop us a line at feedback@swed.org.uk with any questions.

D Changes

27-8-2004
Initial draft.
31-8-2004
First release.
2-9-2004
Fixed minor line spacing problem.