Managing ontologies for modelling the physical and political structure of a university

Sebastian Rahtz

erewhon

Background

Our problem is to come up with a framework in which to model the structure of the University, both political and physical.

- what do we record?
- how do we store it?
- how do we allow retrieval from it?

Remember, in what follows, that Oxford is unusual in having legal entities (the colleges) distinct from the University. Other institutions will have something similar.

Oxpoints I

The first generation of our work, which we called Oxpoints:

- Covered colleges and departments as general-purpose ‘units’
- Modelled a hierarchical arrangement of buildings within units
- Showed time-based data as an attribute of e.g. a college
- Was managed as a large XML file mediated through a series of *ad hoc* XSLT transformations
- Used the Text Encoding Initiative schema

Problems with this approach

- It did not cope with change over time (buildings change hands, departments migrate and amalgamate)
- It did not allow for a building being used by several departments at once
- It did not record any relationships between units; e.g. that a research group is part of a department, or that a rare books library is part of a bigger library
- It was not easily extensible to other data types and relationships in the future (network access ducts? wireless access points? bicycle stands? trees?)

Requirements for Oxpoints II

Time based data	A time dimension should be introduced to be able to create historical maps.
Rich Relationships	It should be able to express many different kind of relationships between entities in the system.
More Entities	Besides colleges and buildings many more entities should be added to the system (e.g. drain covers).
Extendable	Since nobody could pin down where OxPoints was to be going it should be as easy as possible to extend the system in any way imaginable.

The underlying framework: RDF

We conducted research for several months and finally decided to go with an RDF (a standard maintained by the W3C, meant to express relationships between entities) based solution.

Time based data	We have found a way to introduce a time dimension using named graphs
Rich Relationships	RDF is meant to express rich relationsships.
More Entities	RDF has no restrictions on the elements it talks about. As long as you can make up a URI for the element, you can use RDF to describe it.
Extendable	RDF is very generic and meant to be extended.

erewhon

The implementation: Gaboto

Gaboto (`http://gaboto.sf.net`) is an open source generic RDF storage engine that allows for automatic mapping from RDF to Java objects and is able to cope with time in RDF. It was developed for Oxpoints, but is completely generic.

Gaboto gives you RDF's flexibility of storing objects, their properties and relationships between objects while mapping these structures to first class Java objects.

The mapping rules are all configured in an XML configuration file from which the necessary classes are automatically generated. So the basic steps in using Gaboto are:

1. Write configuration file, define objects, their properties and relationships.
2. Execute configuration script, that basically generates a jar file.
3. Plug in the jar file and start writing your system.

erewhon

Ontology examples

```
<ObjectProperty xmlns="http://www.w3.org/2002/07/owl#"
  about="#associatedWith">
  <comment xmlns="http://www.w3.org/2000/01/rdf-schema#"
  >Defines that two entities are associated with one another (N:M)</comment>
</ObjectProperty>
<ObjectProperty xmlns="http://www.w3.org/2002/07/owl#"
  about="#capacity">
  <comment xmlns="http://www.w3.org/2000/01/rdf-schema#"
  >The number of cars that can park</comment>
  <domain xmlns="http://www.w3.org/2000/01/rdf-schema#"
    resource="#Carpark"/>
</ObjectProperty>
<ObjectProperty xmlns="http://www.w3.org/2002/07/owl#"
  about="#occupies">
  <comment xmlns="http://www.w3.org/2000/01/rdf-schema#"
  >Describes that a unit occupies a place</comment>
  <range xmlns="http://www.w3.org/2000/01/rdf-schema#"
    resource="#Place"/>
  <domain xmlns="http://www.w3.org/2000/01/rdf-schema#"
    resource="#Unit"/>
</ObjectProperty>
<Class xmlns="http://www.w3.org/2002/07/owl#"
  about="#Building">
  <subClassOf xmlns="http://www.w3.org/2000/01/rdf-schema#"
    resource="#Place"/>
</Class>
<Class xmlns="http://www.w3.org/2002/07/owl#"
  about="#College">
  <subClassOf xmlns="http://www.w3.org/2000/01/rdf-schema#"
    resource="#Unit"/>
</Class>
```

erewhon

The configuration file (simplified sample)

```
<GabotoEntity name="Unit" type="Unit">
 <properties>
  <property name="address" type="Address" uri="adr"/>
  <property name="homepage" type="Website" uri="hasHomepage"/>
  <property name="OUCSCode" type="String" uri="hasOUCSCode"/>
  <property name="itHomepage" type="Website" uri="hasITHomepage"/>
  <property name="name" type="String" uri="title"/>
  <property
     name="occupiedBuildings"
     type="Building"
     collection="bag"
     uri="occupies">
   <indirectProperty uri="hasLocation" n="2"/>
  </property>
  <property name="primaryPlace" type="Place" uri="primaryPlace">
   <indirectProperty name="location" uri="hasLocation" n="1"/>
  </property>
  <property name="subsetOf" type="Unit" uri="subsetOf">
   <indirectProperty uri="hasLocation" n="3"/>
   <unstoredProperty uri="parent"/>
  </property>
  <property name="weblearn" type="Website" uri="hasWeblearn"/>
  <passiveProperty
     name="hasSubsets"
     type="Unit"
     relationshipType="1:N"
     uri="subsetOf"/>
 </properties>
</GabotoEntity>
```

erewhon

So we move to the second generation of Oxpoints

- Still covers colleges, departments, libraries, museums, and carparks, but also anything else one may encounter
- Models political units, physical buildings and a set of relationships between them
- Is managed in an RDF database
- Has an explicit RDF ontology, using external standards where appropriate (Dublin Core, GML, vCard)
- Explicitly models time and changes

Types of objects in our system

- Building
- Division
- DrainCover
- Entrance
- Faculty
- Group
- Image
- Room
- ServiceDepartment
- Site
- WAP
- Website

erewhon

Types of relationship in our system

- associatedWith
- capacity
- hasITHomepage
- hasLocation
- hasOBNCode
- hasOLISCode
- hasOUCSCode
- hasPrimaryPlace
- hasWeblearn
- homepage
- inImage
- occupies
- physicallyContainedWithin
- subsetOf

Collection

We have collected information on about 1000 ‘political entities’:

Library	743
Unit	19
College	45
Department	186
Museum	7

along with

Carpark	10
building	277
relationships	891
coordinates of points	283

Where did this data come from?

- Our central registration database provided the core structure of units, addresses etc
- Library system provided the catalogue of library codes and names
- Our own nosing around added web page links and photos
- Coordinates were created about 50% from walking around with a GPS, 50% from reading off Google Earth
- Our Estates Department provided a set of their building codes and informal building descriptions

How are things identified?

- Everything has a unique, stable Oxpoints ID (simply a number)
- Registration codes are stored as properties of relevant objects
- Library codes are stored as properties of relevant objects
- Buildings have an optional property of their Estates code
- Objects all have a type

Storage and management

An initializing Gaboto can read data from

- Serialized RDF representation
- A relational database backend for Jena
- A TEI-encoded XML file

The first of these is the backup format.

Editing data is currently a manual task.

Future plans

We intend to manage

- A read-only Gaboto instance, reinitialized every day, to deliver lookup services
- A read/write Gaboto instance linked to a relational database for managing changes and additions
- A map-based interface for adding data

erewhon

Retrieval

The main retrieval mechanism is via a web service which can:

- Select objects by type, by internal ID, and by various codes (from registration, estates, libraries etc)
- Follow relationships (eg from political entity to buildings which it occupies)
- Sort by properties (typically dc:title)
- Serialize the results in a variety of formats

Supported formats

We currently support

xml	Serialized RDF in XML.
json	Serialized as JSON for consumption by Javascript etc (with optional callback function)
kml	Summary information in the KML format used by Google Earth and Google Maps
gjson	The summary form used by KML, serialized as JSON (with optional callback function)

We also currently convert KML to any format supported by *gpsbabel* (support for many GPS and satnav devices)

erewhon

Demonstration

1. Oxford Colleges (using Google Maps): `http://maps.google.co.uk/?q=http://m.ox.ac.uk/oxpoints/colleges.kml`
2. Oxford Museums: `http://m.ox.ac.uk/oxpoints/museums.kml`
3. Balliol College: `http://maps.google.co.uk/?q=http://m.ox.ac.uk/oxpoints/oucs/ball.kml&z=16`
4. Balliol College (RDF): `http://m.ox.ac.uk/oxpoints/oucs/ball.xml`
5. Balliol College (JSON): `http://m.ox.ac.uk/oxpoints/oucs/ball.json`
6. Balliol College: `http://maps.google.co.uk/?q=http://m.ox.ac.uk/oxpoints/oucs/ball.kml&z=16`
7. Balliol College (all buildings): `http://maps.google.co.uk/?q=http://m.ox.ac.uk/oxpoints/oucs/ball/occupies.kml`
8. All Souls College and Ashmolean Museum:

erewhon

Arcana

1. (using old Oxpoints) Map of Oxford in Google Earth showing colleges, and time line `http://www.oucs.ox.ac.uk/oxpoints/colleges.kml`
2. Map of Oxford using Open Streetmap showing university departments `http://www.oucs.ox.ac.uk/oxpoints/test9.html`
3. Map of Oxford showing running route, with background on old Ordnance Survey map. `http://wheresthepath.googlepages.com/wheresthepath.htm`; using GPX data; result is `picture3.png`
4. Oxford University carparks, displayed on TomTom `http://m.ox.ac.uk/oxpoints/Carpark.tomtom`