GSC Hackathon, EMBL-EBI 16-18 January, 2012

Attendees: Travis Harrison (ANL), Chris Hunter (EBI), Phil Jones (EBI), Renzo Kottmann (MPI Bremen), Maxim Scheremetjew (EBI), Wim de Smet (Ghent Univ), Peter Sterk (Univ of Oxford), Doug Wendel (Univ of Colorado)

(Monday 16 January 2012 9:30-10:00) Attendees introduce themselves and describe their interests for this meeting

We'll work in 3 groups: Java programmers, Python programmers and Ontology developers

It is good to have certain user stories e.g.

- want export data in gcdml from db,
- converting a csv (excel) file to gcdml,
- I have a gcdml file, I am a biologist and have an excel...

maybe we agree on a template excel for which we have a XSLT (even with pre-validator). This could be plugged into a web tool that users can upload their files to, or even input the data manually and get back a CSV (or excel) file AND the valid GCDML file describing their data.

Furthermore it would be quite useful for biologist to autogenerate an editable form from uploaded GCDML or CSV files.

We're now discussing whether there are any things in the current schema we're not happy with.

Hard-coded enumerations: it is not practical to hard-code ontological terms into the schema. Peter & Chris will work on the establishment of a GSC CV for new terms during this hackathon.

Units: free-text units are not parsible by software. Phil recommends the use of the OBO unit ontology, but we need to make sure it fulfils our requirements. We need 1. the name of the ontology, 2. its version, 3. its identifiers and 4. the names of the terms to curate data.

We all agree we will use *a* unit ontology (whichever one fulfils our requirements) and we do not want to have exceptions. What about new units? We need to be able to have an escape to capture those. We would prefer to only allow SI units, the question is how to enforce this.

- PSI Validator can be found here: http://www.psidev.info/index.php?q=node/336
    - Publication on smeantic validator: http://onlinelibrary.wiley.com/doi/10.1002/pmic.200700064/abstract
    - Schemas: sourcesList: http://www.psidev.info/files/validator/CvSourceList.xsd

    - CV Mapping http://www.psidev.info/files/validator/CvMapping.xsd
- PSI-MI - Probably the most mature PSI format. http://psidev.info/index.php?q=node/31

Chris and Peter are looking at the various unit ontologies out there.

- OBO: unit ontology: One of the more complete ontologies. However, still incomplete. Defines derived units separately (e.g. gram - kilogram, milligram). A better approach would be to have the prefix + unit, e.g. kilo + gram would make it easy to convert between gram and kilogram. Concentrations are explicitly defined. I'd prefer a different approach: [concentration] + [mass/quantity unit] + [volume unit]. This is far more flexible as not every possible combination needs to be defined.

- Morfeo/MUO: less complete, but it seems to have a more systematic approach and implements (part of) ucum, which defines a unified code for units of measure with the purpose of facilitating unambiguous electronic communication.

- Sweet : appears to be work in progress, not yet got any definitions in place, but appears to have most of the base units (as defined by SI) and a higher level class to describe the prefix to those terms (e.g. micro, kilo, pico etc...)

- QUDT units ontology (The NASA Quantity - Unit - Dimension - Type Ontology)- seems to be quite complete, but still lacking in definitions, and the SI units class seems to be deprecated?! -- that's because the individual items have been moved into more specific classes within the hierarchy. Bonus is that the OWL file contains the conversion details too.
- MGED also include all the common units within their ontology, it seems fairly complete and has definitions, but doesn't include the conversions, might be useful if we want to make use of MGED ontology for other fields as well as units?

Other stuff we've come across: ucum (described under Morfeo) and a java API units of measure API

Alternatives to ontologies

- unitsML (XML schema). Systematic approach to representing units. Native support for unit conversions.

Conclusions: The easiest one to use is in our opinion the OBO unit ontology. Although not complete, we can supplement missing units/terms and work with the ontology people to get terms we need incorporated. Communication with OBO UO ontologie developers may be more straightforward than with developers we have no 'relation' with.

There are other solutions that have a number of interesting features. First a more structured approach to describing units, derived units etc. Some have support for conversions. It would be more tedious to use those for curation, though. For example, if you wanted to describe a concentration, you would have to enter all base units + any prefixes separately, a task you probably don't want to leave to most biologists.

Programmers tasks:

Create a generic GCDML web application

Use cases:

- Build or edit a valid GCDML file
    - By typing into a form
    - Upload or download a CSV file (etc.)
- Validate existing GCDML file (see Markup Validation service)
    - schema validation with full error report
- Convert from version x->y

Functional requirments:

- Deploy anywhere
    - easy to skin with CSS
- Apache 2.0 licence

Java bindings:

We will target java SE6 as minimum required for the bindings (java SE5 is the minimum required by JAXB).
If files are generated with the current xjc compiler, then bindings should be compatible with java SE5 up, but they will be compiled with code target 1.6.

One of the issues found is that there are two class hierarchies intermingled, one for minimal MixS compliant files, and then an extension on top of that. As the extension is also MixS compliant, it's probable that users will usually stick to these. (any file with a "MIGS" report is also a valid file if you change the MIGS -> GCD).

JAXB bindings to-do list:

- Certain measurements like "pregnancy" aren't a descendant from the Measurement types for XML schema reasons. A superclass or interface collecting these might work, or just map them directly in classes

- Certain names with underscores might be renamed
- Remove the deprecated “submitTo...” elements
- Sample location does not make use of GML yet
- Adding javadoc, generating and publishing javadoc

The following workflow is used for binding development:
- jaxb classes can be generated through an ant task, regenerated whenever the XML files etc change. Customisation should happen as much as possible through the XJB files.
- Utility classes that use these files will live in their own subdirectory (jaxb) of the repository for now. These might or might not include a complete facade on top of the generated files (in the form of interfaces).
- ant-targets can be provided to compile everything and roll it into a jar, as well as create documentation.

It would be nice if the generated classes could at least be split up into packages, but unfortunately (after some research), it appears that you can only set one single package for everything. It might be possible to get more control with annotations inside the schema, but we do not want to pollute the schema with those. The only other option is breaking up types into different (sub)namespaces, which would map to different packages.

Packages for JAXB bindings:
- reports
- measurements
- sequencing
- isolates
- environments (water, soil, …)
- …?

We mostly agree that users will require a domain model anyway, so the best way to get the bindings working is building a domain model (tedious but doable) and mapping it to the JAXB API (need to think about entry/exit points. would like to make it possible to just construct a full object tree, but we’ll need to validate it entirely when serializing. The easiest way to go into it programmatically would be a builder API. How do we deal with the simplest and most common use case: users simply have some data in CSV and want to convert to GCDML? → see also the discusson on a web form interface)

The major design decision is how close the domain model follows the XML schema, and whether it supports all possibilities (especially both MIGSReports and GCDReports, or just the latter one).

Ways of keeping the class explosion down: for instance keep Measurement as a name field with the attributes as its other fields. There are a few special cases, but they can either be solved with this class or with a single extra inheritance step.

XML Scham and JAXB processing based Versioning

http://www.funkypeople.biz/knowledge/JavaXml-v2.pdf

GSC DB

located at https://gcdml.gensc.org/wiki/GscDb

NOTE: The file gsc_mixs-2.1.dump in the tar archive on this page contains a bug. Before you install the database (described in this document for Mac OS X and linux), make the following correction:

Under
CREATE VIEW env_checklists AS
.....
'select gcdml_name from gsc_db.environments order by 1'::text) .....

correct the schema name in this select statement from gsc_db to gsc_db_2_1:
'select gcdml_name from gsc_db_2_1.environments order by 1'::text)

Python bindings:

pyxb: http://pyxb.sourceforge.net
install pyxb

1. create python package from gcdml schema:
   pyxbgen -u https://gcdml.gensc.org/svn/schema/gcdml/trunk/base/gcdml.xsd -m /yourdirpath/gcdmlSchema

2. use binding packing to validate and load xml into schema:
   import gcdmlSchema
   xml = file('afile.xml').read()
   data = gcdmlSchema.CreateFromDocument(xml)
If no errors, is valid

example files: https://gcdml.gensc.org/svn/schema/gcdml/trunk/test-doc/2.0/

USE of GCDML for files :

it is not well documented how exactly best use NasReports and MIXS/GCDReports and then the specific checklist reports like metagenome, prokaryote etc. Need to clarify exactly what we mean by “study”, “report”, … (any other terms?)
To explain from top to bottom:

- NasReports is just a container to technically put together: many MIXS or GCD Reports
- The difference between MIXS or GCD Reports is that MIXS Reports only implement the checklists and nothing else but the checklists; where as GCD REports are a superset of MIXSReports and allow additons and extension beyond what is defined in the checklists
- inside a GCDReports element one can assemble all different reports originating from a single study. Therefore a GCDREports element has as direct child element a <studyData> element which contains metadata about project managment aspects like e.g. who is the PI email contact etc the one or many concrete checklists reports can follow which should belong to a particular study

Posssible element name change could be GCDReports to GCDStudyReport(s)

It should be made clear that NasReports is the only element that can be used at the root level in a GCDML file, and then have the validator explicitly check for this too.

Installation of **GSC MIxS DB, A database for the MIxS checklists and environmental package, in** PostgreSQL 9.1 on Mac OS X lion (Peter)

(largely according to
http://www.jonathandean.com/2011/08/postgresql-8-4-on-mac-os-x-10-7-lion/)

You may need to change ‘9.1’ in several places below if your postgreSQL version is different from 9.1.

Mac OS X lion has by default a user _postgres, which causes problems when you install a new version of postgres. Removed this user first as follows:

$ sudo dscl . delete /Users/_postgres

Create a new user ‘postgres’ using either Users & groups in System Preferences (no admin rights needed for user ‘postgres’) or do it from the command line:

sudo dscl . -create /Users/postgres UserShell /usr/bin/false

Download the latest postgres image from
http://www.enterprisedb.com/products-services-training/pgdownload

Add the following line to .profile (or .bashrc):
export PATH=/Library/PostgreSQL/9.1/bin:$PATH

Type source .profile (or source .bashrc)

Install PostgreSQL using the installer in the downloaded image.

Change ownership of the data directory:
$ sudo chown postgres /Library/PostgreSQL/9.1/data/

And then initialize it:

	$ sudo -u postgres initdb -D /Library/PostgreSQL/9.1/data

You can now start the database server using:

	$ sudo -u postgres postgres -D /Library/PostgreSQL/9.1/data

or

	$ sudo -u postgres pg_ctl -D /Library/PostgreSQL/9.1/data -l logfile start

Create a new database called ‘gsc_mixs’:

$ sudo -u postgres createdb gsc_mixs

Download the tar archive following the link on https://gcdml.gensc.org/wiki/GscDb .

In a terminal, cd to the gsc_mixs-2.1 directory (make sure it is readable for all users, e.g. chmod 755 gsc_mixs-2.1) and create a new role (gsc_mixs_role):

$ sudo -u postgres psql gsc_mixs < create_gsc_mixs_role.sql

Create extension tablefunc:

$ sudo -u postgres psql -d gsc_mixs
Password:
psql (9.1.2)
Type "help" for help.

gsc_mixs=# create extension tablefunc;
CREATE EXTENSION

```
Create language extension plpgsql:
# At psql prompt:
create language plpgsql;
```

Return to terminal (\q) and load the dump file (cd to the gsc_mixs-2.1 directory first):

$ sudo -u postgres psql gsc_mixs < gsc_mixs-2.1.dump

Installing the database on Ubuntu (probably debian too) (Phil/Maxim)

```
# Retrieve the mixs files:
mkdir gsc_mixs
cd gsc_mixs
wget https://gcdml.gensc.org/raw-attachment/wiki/GscDb/gsc_mixs-2.1.tar.gz
tar -xvzf gsc_mixs-2.1.tar.gz
cd ../
chmod -R 777 gsc_mixs
cd gsc_mixs

# Install PostgreSQL

sudo apt-get install postgresql postgresql-contrib

# Create the required role
sudo -u postgres psql postgres < .gsc_mixs-2.1/create_gsc_mixs_role.sql

# Create language extension plpgsql
sudo -u postgres psql -d gsc_mixs
# At psql prompt:
create language plpgsql;
\q

# Import the database
sudo -u postgres psql -f ./gsc_mixs-2.1/create_gsc_mixs.sql
sudo -u postgres psql -f ./gsc_mixs-2.1/gsc_mixs-2.1.dump -d gsc_mixs
```

Side learning: https://research.microsoft.com/en-us/collaboration/fourthparadigm/

General Web Form Development

- HTML, CSS, JS only implementation (this is new nobody has) interacting with web services

- It is very important to help the user to get the most effective way of pre-filling and re-using filled out data: default values where possible
- A great way would be to start off from the Quiime code base

Agreement: we need tools for:
1. web based input forms (start from qime code?)
2. query the ontologies services like e.g. OLS and / or BioPortal (if they have/add versioning) we need to get the data locally
3. web service for the terms in the GCDML latest version or from certain version
4. Some terms will not be available in any of the ontologies at a quick turn-around. A GSC controlled vocabulary can be supplied to allow the user to enter some of these terms. Additions could be managed by a user feedback process (after a lot of UI based measures to discourage them).

For (3) you will need a local version of the GSC database. Will we then put in this database the controlled vocabulary, list of ontologies and...?

Point of contention: OLS has no support for ontology versioning, this is why we would need our own services. We need versioning in the ontology for several reasons. The first is that, although they shouldn't, some terms get deleted and some identifiers do change. The second is that we will use the ontology for semantic validation, which requires a fixed ontology to work.
The BioPortal might be another option, cause they support versioning.

Code Development - General Points
Apache license
Move to github
Java
Package root **org.gensc**