Prestudy

Moving Reportnet forward

within the context of SEIS

Discussion of technical options

Prepared by

Søren Roug

December 2008

Version 2.0

European Environment Agency

Version management

No	Date	Changes	Author
1.0	30.07.2007	Initial version	Søren Roug
1.1	05.08.2007	Added example of Norway's integration into SEIS.	Søren Roug
2.0	01.10.2007	Updated with conclusions from Reportnet sprint 17-21 September	Søren Roug

Contents

1. What are we trying to do?

The purpose of Reportnet is to support the European environmental reporting. We have two types of stakeholders; the providers of data and the requesters of the data. Between them there is a pipeline consisting of: Delivery, QA, Aggregation and Analysis.

When developing Reportnet we have until now focused on the first two segments. The providers know what to deliver, when and in what format. The requesters get automatic quality analysis and a feedback channel back to the provider.

This is not enough, because it targets only the data that is agreed between international institutions and countries. Within the context of SEIS there is a need to handle data from many other information providers and Reportnet – as the EEA/Eionet reporting system – needs to be further developed for that purpose.

Therefore we must revisit the design to also handle *collection of data*, as it is one of SEIS' core principles that data should be managed as close as possible to its source. And we must not forget to make the rest of the pipeline more efficient either.

We will attempt to increase the speed of manual QA by making it easier to document the QA steps and reuse the supporting tools for future delivery cycles.

The Aggregation today is typically done manually with ad hoc tools such as Excel and MS-Access. If we look into our XML deliveries, we have a distinct lack of tools. The problem is that the aggregation is often not repeatable, because the operator didn't write down what he did to clean the data, how outliers were handled or gabs were filled. An analogy to the system we want to build is the what-if calculation people do in a spreadsheet. The formulas are fixed and the variables are the data the member countries deliver periodically.

And finally we want to make the resulting information easier to use by providing it in the format the user needs, and integrate Reportnet into other systems such as Inspire and the IMS at EEA.

1.1. Definitions, acronyms and abbreviations

Term	Description
Shapefile	The ESRI shapefile is a popular geospatial vector data format for geographic information systems software. It is developed and regulated by ESRI as a (mostly) open specification for data interoperability among ESRI and other software products.[1] A "shapefile" commonly refers to a collection of files with ".shp", ".shx", ".dbf", and other extensions on a common prefix name (e.g., "lakes.*").
Feature	Shapefiles contain 'features'. A feature is a point, line or polygon that depicts e.g. a hill. The feature can have 'attributes'. For a hill the attributes could be height and name.
QA	Quality Analysis
GML	The Geography Markup Language (GML) is the XML grammar defined by the Open Geospatial Consortium (OGC) to express geographical features.
SEIS	Shared Environmental Information System in Europe

IMS	Indicator Management System
LAEA5210	Lambert Azimuthal Equal Area 52ºN 10ºE. Recommended for geographical vector data that are only intended to be used in production of small-scale maps for reports.
CDR	Central Data Repository (http://cdr.eionet.europa.eu)
AWK	A programming language designed for processing text-based data
Python	A general purpose object-oriented programming language used in Zope
AJAX	Asynchronous JavaScript and XML, a technique used in web application development. Provides more responsive interaction

1.2. Definition of roles

To better understand how a delivery changes hands in its life cycle, we must define some user roles.

Role	Description
Data provider	The data provider is anybody who makes a delivery to either CDR or another repository. It is typically an employee of an NFP, but can also be an employee of the secretariat of a commission. The term *Reporter* is used as a synonym.
Requester	The requester is the organisation that has requested the data. This is typically EEA, Eurostat, and OECD etc. When the data are received, the requester aggregates and analyses them.
QA worker	A person at the requester organisation, who does QA on a delivery.

2. Work done in 2004-2007

For those who haven't followed Reportnet developments closely in the past years, let's take a moment to recapitulate the development work done on Reportnet.

2.1. New features in 2004

In 2004 we implemented the GDEM (Generic Data Exchange Mechanisms) modules. These were converters, webforms, automatic QA and workflow. As CDR is written in Python and the necessary modules only existed in Java, we implemented all modules except workflow as web services using XML-RPC to contact them.

Since the start CDR has had something called *quickviews*. The purpose is to e.g. show a spreadsheet as an HTML table thereby making it faster to check the content. With the preparations for XML deliveries, we realised there is no native application to show an XML file, so we created an new conversion service using XSLT scripts to render the XML files as an HTML page, MS-Excel spreadsheet etc. These scripts must be written manually for that particular XML format.

We noticed that a large number of dataflows handle relatively small amounts of data. For instance there a directives that require periodic *implementation reports*. These are questionnaires having about 50 questions. Other dataflows are reports of new measuring stations being installed or removed. The webform system was developed to make it easy to implement a new questionnaire.

Since XML is a self-describing file format, we can do something that can't be done on spreadsheets. We can write scripts that check whether the content of an element in an XML file is within given constraints. We can summarize all values an run many other checks. The purpose is to free the requester from the mundane QA work and let him concentrate on the tasks that require human intelligence. The workflow system detects what XML files are available in the envelope, and runs the correct scripts on them. The result is called a feedback report and it is stored in the envelope as a permanent comment on the delivery.

The workflow system is used to connect all the activities together. Before, the workflow was hard-wired. The data reporter created an envelope, uploaded files and released it. Now we can execute additional steps such as run the QA scripts automatically.

2.2. New features in 2005

In 2005 we implemented Unified Notification Service (UNS), automatic conversions, single-sign on and data merge module – an early experience relevant for the QAW to be discussed later.

The rationale behind the Unified Notification Service is that there are many things happening on Reportnet systems that are of potential interest to others than the author. An example of an event could be an approaching deadline for a dataflow. Since Reportnet is deployed over several websites, we decided it was an unnecessary complication for each individual website to keep track of user's emails and subscriptions. Therefore we implemented UNS as a service. The user subscribes to notifications in one central location. Whenever something happens on the Reportnet sites worthy an event, the site sends one copy of the notification to UNS, which then delivers it to the people subscribed to the channel.

The automatically conversions are an extension to the converters developed for GDEM. They use information retrieved from DD to generate six standard conversions for XML formats described in DD.

With this network of websites, that Reportnet really is, we have an authentication issue when one website uses another website to get a service performed on behalf of the user. An example is when the users uses the conversion service to retrieve and convert a file stored in an unreleased envelope on CDR. Since CDR is part of Reportnet, it is a trusted site, but this is not always the case. The Central Authentication Service (JA-SIG CAS) solves this problem, and we have implemented it on DD and ROD currently.

The Data Merge Module or DMM is an application to easily aggregate deliveries from several countries. It is an MS-Windows application that asks you what dataflow and period you want to work on, then it queries CDR for the available envelopes and aggregates the XML files into a database table in several formats.

2.3. New features in 2006

The year 2006 was mainly used to prepare for the Habitats Directive Article 17 dataflow. In this dataflow, the data providers create XML-based factsheets for every habitat type and species needing protection. This amounts to about 100 of each. Each factsheet can have up to four GIS attachments.

The new functionality developed was:

- Conversion of ESRI shapefiles to GML.
- Conversion of GML to ESRI shapefiles.
- Conversion of GML to GoogleEarth KML files (only if projection is LAEA5210).
- Displaying GML files as PNG with and without background maps.
- Displaying GML as Flash, where the user can pan, zoom and inspect attributes.
- QA for GML files. GML files can become very large, so XQuery scripts were determined to be inadequate. The QA is written in Python. The checks are only run if the projection is LAEA5210.
- A webbased GIS editor written in Macromedia Flash.
- Web questionnaires, QA rules, stylesheets and file formats for Article 17.
- A large number of modifications to the workflow for the Article 17 dataflow.

It needs to be mentioned that for the type of GIS information we handle, the only way to check that the features[1] are a the correct location, is to show them on top of a map of Europe. These maps were retrieved from the EEA map server using ArcXML.

2.4. Special dataflows implemented

These dataflows are the ones that get special treatment in Reportnet.

End-of-life for vehicles

This is an implementation report done as a webform for the Topic Centre on Resource and Waste Management.

Monthly/Summer Ozone

Every month the member countries must report on occurrences where the ozone level goes above a threshold. At the end of the season they must provide a summary of the summer's occurrences. The dataflow is implemented as a set of webforms.

WISE Article 8

The WRc was given the contract to implement the Article 8 dataflow. The object is to collect information on groundwater and surface water monitoring stations and monitoring programmes. It is implemented as an MS-Windows application that uploads XML files to CDR, where they are sent through the normal QA system.

Habitats directive Article 17

Every six years the member countries draw up a report on the implementation of the measures taken under the Habitats directive. The report takes the form of XML-based factsheets on habitat types and species. These are implemented as webforms.

Groundwater bodies

[1] A 'feature' is a point, line or polygon. A shapefile contains many features.

This was the first dataflow to implement with webforms. The purpose is to update the information on groundwater bodies. This means the user loads last year's data, then modifies it for this year's delivery.

Rivers

Once a year the countries report information on rivers. They supply data in XML, Excel, tab-separated ASCII text or Access database format. If in MS-Excel (the most common), the spreadsheet is converted to XML and then sent through the automatic QA system. The conversion is only possible because the MS-Excel format is defined in DD.

3. How is QA and aggregation done?

The purpose of this section is to describe actual user procedures in order to ensure the proposed new systems addresses the right requirements.

3.1. Hermann's experiences with WISE article 8

1. Check status of deliveries

   I first checked the Art.8 deliveries through a normal CDR search: http://cdr.eionet.europa.eu/resultsdataflow?dataflow_uris=http%3A%2F%2Frod.eionet.eu.int%2Fobligations%2F520&years%3Aint%3Aignore_empty=&partofyear=&country=&sort_on=reportingdate&sort_order=reverse

   The available search results are not very helpful for me as the Reportnet QA worker. There is nothing about the release status of the envelopes and no indication about the actual files inside them. So I had to invent my own search with more informative search results.

2. Write your own CDR search

   This is the result of my quick-hack programming: http://cdr.eionet.europa.eu/recent_etc?RA_ID=520 I will probably not win a web design award for it, but this page tells me considerably more about the deliveries than the regular CDR search results.

3. Contact reporters that have draft envelopes in CDR

   a) Search for draft envelopes in the above list
   b) Go to the envelope
   c) Go to its history page
   d) Grab the user name of the data reporter
   e) Go to Eionet Portal
   f) Search for the user's email address
   g) Write him/her a mail asking why the envelope is still in draft status

   The typical result was that they had forgotten to release the envelope. So I did it for them (to speed up the process.)

4. Finish released envelopes

   While waiting for the replies from step 3, I used the list from step 2 and went manually into all Released envelopes and completed them, in order to make sure that the data reporter wouldn't bring them back to draft mode and change the data after I had made my downloads. (As far as I know: there is no Reportnet/Workflow functionality that would help me doing this crucial step).

5. Fix schema location of XML files

   While going through the envelopes and doing step 4, we saw that many XML files had missing values for the schema location. This means that no CDR XML file search will ever find them. As there is no Reportnet functionality to search for XML files without schema location, we had to manually go through the list of envelopes, and fix the missing XML schema location value. I think it was for more than 100 files.

6. What about envelopes with wrong/missing Obligation tag?

   The CDR searches in steps 1 and 2 are based on the assumption that the Obligation tag is correct. But what if not? I hence manually went through the CDR folder hierarchy of all country with missing data, in order to make sure that I wouldn't overlook data uploaded under a wrong obligation (that the CDR searches above would hence not find).

7. Download XML files

   Now all envelopes are in "Complete" status. We also know that all XML files have the correct schema location value, so we can download the files. Søren first helped my out with his small XML-RPC request. But in order not to bother him all the time, I wrote my own quick-hack page that lists all XML deliveries with a certain format (i.e. schema location), which I used to retrieve files via a Linux tool called wget:

   http://cdr.eionet.europa.eu/resultsxml.txt?xml_schema_location=http%3A%2F%2Fwater.eionet.europa.eu%2Fschemas%2Fdir200060ec%2FGroundWaterMonitoringStations.xsd&years%3Aint%3Aignore_empty=&partofyear=&country=&sort_on=reportingdate&sort_order=reverse

   What about zipped XML files?

   My search above wouldn't find zipped XML files, as zip files don't have a schema location. So I went back to the list generated under step 2 and indeed: I found 7 zip files from Ireland and downloaded them manually.

8. What is in all these 300+ AutomaticQA reports?

   Yet another sad story of manual work for the CDR QA worker. I made my own quick-hack download list of all 300+ AutomaticQA reports and downloaded the files via wget. Then I grepped through the QA report text for "Success" or "Not successful" or similar. The result was that all but one reports were saying that there are no errors in the files. That sounded really good. However, a simple XML schema validation with a non-Reportnet tool resulted in 54000+ schema validation errors, obviously overlooked by the Reportnet AutomaticQA.

9. Converted everything to CSV with XSLT

10. Used reference tables also in CSV, and used AWK as the scripting language.

After all the steps above, 2 weeks had passed and I was eventually able to actually start the QA work, i.e. divide the 54000+ errors into negligible and relevant errors and contact countries to clarify the latter ones. From my description above, it is perhaps clear that rather than a QA workbench (as proposed in section 5.6) , I see a need for smaller enhancements around the CDR workflow that would make (some of) the manual steps obsolete.

3.2. Søren's experiences with article 17

1. Extracted the QA errors from the feedback reports with a stylesheet and imported it into a table.
2. Extracted a list of files in the envelope with a stylesheet and imported it into a table.
3. Extracted the events that happened on the envelope. These are workflow events, file uploads, etc.
4. Converted all XML factsheets into database records. Since the factsheets are designed for convenient reporting, they are quite complex. They have repeating sections inside repeating sections and converting one factsheet type into a dataset structure creates about 15 tables.
5. From the database I extracted all the texts and sent them to be translated. The factsheets have an attribute saying which language the reporter used, and extracting the Swedish texts to send to the Swedish translator was easy.
6. The ETC wanted all the GML files to be converted to ESRI shapefiles (sic). Otherwise they had trouble handling the amount of files. But what they also wanted, was to have some attributes added to all features during the conversion. For habitat types it is country, map type, habitat code and the upload timestamp. It makes it possible for them to import all GIS files, and then make a selection to show all habitats in Europe of type 1130. To do it, I modified the GML to ESRI conversion script developed in the ReportnetGIS project.
7. Modified the workflow to allow reruns of the automatic QA. But since the ETC can't do the extraction of the QA errors into the database on demand themselves, they wait until the next day for the results.
8. From here on, the ETC continued with the QA using only the database.

4. Lessons learned

4.1. Automatic quality analysis

It was originally expected that the QA scripts would be an aid to the manual QA, and therefore relatively simple. They would check that all mandatory fields are filled out and codes conform to the requirements. What happened was that we implemented a button making it possible for the data provider to test their files *before* delivery. This made the data requesters realise they could get better *initial* deliveries by putting as many tests into the QA scripts as possible and hence the scripts became more and more complex, making more queries into ever larger tables of station codes and other lookup tables. Some of the scripts now have 1600 lines.

4.2. Manual quality analysis

The ETC/BD was tasked to do the quality assessment of the art. 17 dataflow. Since there were many hundreds of feedback items per country, it was seen as too cumbersome to read all these to see if there was any instances where the reporter had delivered bad information. We therefore implemented the feedback in such a form that the essential information could be machine-read and loaded into a database table.

The feedback is supposed to be humanly readable, because it is essentially a check the data provider can do to verify he is delivering correct data. The QA system generates an HTML page the user can view directly in his webbrowser. What we did to make it machine-readable, was to take an idea from microformats.org. Microformats is the use of HTML attributes to express semantics. Programs can extract meaning from a web page that is marked up with one or more microformats.

We created our own microformat explicitly to describe QA information. With a simple XSL stylesheet, we can the extract the error messages and load them into a database. The data requester can then use normal SQL to find all errors of the same type.

Examples

These examples use the Monthly ozone xml files. The first section is a snippet of a QA feedback report. Converting the HTML with a stylesheet…

```
<div>The following red coloured values are wrong: </div>
<table border="1" error="These values are wrong">
  <tr>
    <th>EoI station code</th>
    <th>Local station code</th>
    <th>Type of station O3</th>
    <th>Type of station EoI</th>
    <th>Type of area EoI</th>
  </tr>
  <tr align="left" key="LI0001A">
    <td>
      <b>LI0001A</b>
    </td>
    <td>VAD</td>
    <td>
      <div>R</div>
    </td>
    <td element="dd208:Station_type_EoI" style="color:red">Traffic</td>
    <td element="dd208:Area_type_EoI" style="color:red">Rural</td>
 </tr>
</table>
```

… will generate the following information:

Key	Field	Error	Value
LI0001A	dd208:Station_type_EoI	These values are wrong	Traffic
LI0001A	dd208:Area_type_EoI	These values are wrong	Rural

and:

```
<div>The following values are not listed in the stations list: </div>
```

```
        <table border="1" error="not listed in the stations list">
          <tr>
            <th>EoI station code</th>
            <th>Local station code</th>
          </tr>
          <tr align="right" key="41B011">
            <td style="color:red">
              <b element="dd208:EoI_station_code">41B011</b>
            </td>
            <td element="dd208:Local_station_code">Dardanelles</td>
          </tr>
```
has the result:

41B011	dd208:EoI_station_code	not listed in the stations list	41B011
41B011	dd208:Local_station_code	not listed in the stations list	Dardanelles

4.2.1. Accepted flag

The accepted flag was also implemented in response to the needs of the Art. 17 evaluation. The ETC/BD wants to send some of the delivered files back to the reporters for redelivery, while at the same time going ahead with those files having no errors.

The reporters will not be able to delete or modify the accepted files.

4.3. Handling GIS data

Our first taste of visual inspection of GIS data occurred, when we added a simple conversion of EPER facilities to GoogleEarth KML files. The EPER deliveries are XML files containing factories with their lat/long coordinates in decimal degrees. Since KML is also an XML format and it also uses decimal degrees, it was a trivial task. The ability to inspect the coordinates of the facilities made it easy (and quick) to verify that no mistakes are made. You can even zoom so far in that you can see whether there is a factory at the location or not.

Therefore, if visual inspection of simple coordinates in XML files are so useful, couldn’t we do the same for real GIS data?

Hence the preparations for Article 17 included the ability to do a better handling of GIS data. Before, ESRI shapefiles were considered opaque and they were ignored in the envelope.

What we wanted to do was for CDR to handle reporting, validation, and presentation of geographical information. It boils down to:

- CDR must verify that the collection of ESRI shapefiles is complete and not corrupted when the reporter uploads
- The reporter must be able to inspect the uploaded GIS files. This would be done by making a *quickview* e.g. an image that can be shown in the webbrowser.
- It must be possible to create QA scripts that test the content of the GIS files. One example could be that the delivered coordinates are within the country’s boundary.

In addition, we developed a Flash based map editor. The assumption was that not all reporters would have GIS software, so rather than receive the maps on paper, we wanted to make it possible to deliver having just a webbrowser.

Given these requirements, the contractor decided to handle all GIS data in GML format. GML is a new vendor-neutral geographical markup language specified in XML by the Open Geospatial Consortium, Inc.® (OGC).

Converting data from one format (shapefile) to another (GML) always risks dataloss if the target format doesn't match in capability. The problem is that this is not a straight forward conversion. GML can't stand alone as a format. It needs an additional XML Application Schema to describe the attributes to add to each feature[2]. If the attribute isn't anticipated then it won't fit in the GML file.

On the other hand, having all the GIS information in one GML file makes it much easier to handle as a unit and convert to e.g. KML. That's a plus.

So why was GML chosen?

One might think that the ability to inspect the GIS files by converting to an image could only be done if everything is in one (XML) file, but when you show as an image, you are not interested in the attributes to the features. You want to show all features. And this means you only need one file: The .shp file. This simple product can show the features using only the .shp file: http://www.qarah.com/shapeviewer/.

However, if you want to inspect the attributes on a feature, you need more than just the .shp file. You also need the .dbf and potentially the .shx file containing the index. If you need to show the GIS information with Europe in the background you additionally need the .prj file. Otherwise it won't be able to place the features correctly on the map. The .prj file contains the projection. Countries close to equator uses a different projection than countries near the north pole. Otherwise they look distorted. There a hundreds of projections. So it isn't enough for a map-viewer to know the projection. It must also understand it and reproject the features to match the background map.

The ReportnetGIS project delivered several types of viewers. A conversion to image showing all features without attributes. A Flash-based viewer allowing to inspect the attributes and a converter to KML. The two first types did not do any reprojection. You could only see Europe in the background if the file was in LAEA5210. The KML converter only reprojected LAEA5210 to WGS84 used by GoogleEarth. Therefore, the presence of the .prj file was immaterial for the visualisation. The converter could not use it.

4.3.1. Article 17 use of GML

The article 17 GML Application Schema doesn't expect any attributes. If a polygon is in the GIS file, it is assumed the species is present. We disseminated the application schema to the countries, requiring them to use the file when exporting from their GIS systems. If they had additional attributes on their shapes, the attributes would not be present in the file uploaded to CDR.

It had both positive and negative consequences. It would not be possible for a country to create one large shapefile containing all the features for all habitat types where each feature

[2] A 'feature' is a point, line or polygon. A shapefile contains many features.

is tagged with the habitat code. This could have saved some time for the country. On the other hand, it also prevented the countries to invent their own attributes that the requesters then had to untangle and this made it possible to automatically show the distribution of a species, because that were the only features in the relevant GIS file.

4.4. Aggregation

The data merging module (DMM) was designed and developed as a mechanism to aggregate XML files from a delivery. The user could in an interactive session query CDR for deliveries for a dataflow, period and country. He would get a list of results. In case a country has delivered twice, he must decide which one is the correct one. Then he downloads and stores the result in a relational database.

When we designed the module we assumed that most new dataflows would be described in DataDictionary. The DMM would query DD to get the table structure, and make the extraction simpler. This didn't happen. Some of the most important dataflows are defined with XML schema tools. E.g. *Wise Art.8*, *Article 15* and *EPER*. With its dependence on DD, DMM can't aggregate these deliveries, and DMM isn't currently in use.

Also, it is an MS-Windows application, which means it has to be installed on the user's PC with all the issues of configuring and distributing new versions.

What we do instead, is to download all files in a delivery manually. We have developed some scripts that can get all files of the same type in one batch operation. We then write XSL stylesheets to make a tree-structured XML file into something more convenient for database work. As you have seen from the description of the experiences, Søren prefers to make SQL insert statements directly, while Hermann and David prefer tab-separated files.

5. Proposals for 2008

5.1. Improvements to GIS reception

How much do we want to restrict the user in delivering GIS?

I recommend to not convert ESRI shapefiles to GML. Instead, we should create quickviews for the individual ESRI and MapInfo files.

.prj – parse the projection information and present it as a HTML page

.shp – show the features on a PNG map

.dbf – show the attributes as an HTML table

.shp.xml – show the metadata as HTML

Since these files are not XML, they will use the original quickview functionality, which calls a subprogram to display the file in a different format. As the only difference between the files in the shapefile is the suffix, the converter can figure out what the other files are called, and use them in the display.

The online GIS editor is quite impressive, but it is too dedicated to Article 17 needs.

First of all, you can only report grid cells in 1, 10 and 100 km units. Very few dataflows has this need. Most deliver points and polygons – e.g. area boundaries of sites.

Secondly, since it read and writes GML, for each dataflow you want to use it for, you must create an XML schema containing definitions of the attributes allowed on the features.

I recommend that we scrap the online editor. If this is too drastic a measure, we should change it to load from/save to CDR in a simpler format that CDR then automatically converts to shapefile. But remember, we will need to implement delivery of points and polygons as well.

The online editor is also used for the Flash based viewer. It can show all types of features, and you can view the attributes on individual features, but only if the projection is LAEA5210 will you see Europe in the background. The viewer is quite useful, especially if we can extend it to only show those features with the same attributes values – a filtering mechanism. We recommend we keep the Flash viewer.

The Article 17 dataflow converted shapefiles to GML, silently dropping all unrecognised attributes in the process. If we are not converting to GML any more, we propose to create QA scripts that check for the presence of the correct attributes. The QA algorithm would be like this:

> For dataflow *x*, all .dbf files must have the attributes *y* and *z*, optionally *r*, *s* and *t*, and no other.
>
> Warn if there is an unknown projection in the .prj file.

The current GIS QA also operates on the GML. It checks that the country has delivered its polygons inside its own geographical area. Since it is written in Python it doesn't need the file to be in XML. We recommend to reimplement it to work directly on shapefiles, and know the bounding boxes for all the commonly used projections – not just LAEA5210.

5.2. CDR reimplementation

As demonstrated in the previous chapters, expectations about what CDR should be able to do has grown over the years. CDR has become a victim of its own success. If CDR is going to handle the increased requirements, we will need better file storage, better converter integration, better webforms, and since both converters and webforms are already implemented in Java, it means making the rest Java too.

5.2.1. Reprogram CDR into Java

In essence, CDR is a file server. Therefore the first task is to find an object database, that can provide the same navigational user experience of collections, envelopes and files. These object must have access controls lists to prevent people from destroying each other's data. When choosing the foundation blocks, we'll have to keep in mind the requirements of the more advanced features. For instance, to use XQuery in large scale, the XML files in a delivery should be saved in an native XML database. The eXist product could be a solution, but it doesn't have a workflow system nor the ability to store attributes with its containers. Only title and permissions. Alfresco is another contender. It has most of the features, but perhaps it is so focused on groupware that it is difficult to refocus it.

Features to reimplement:

- Collections, envelopes, files and feedback items.
- Access control lists.
- Document centric workflow. Look at Alfresco, and Java best practices.
- Workflow operation pages.
- Scripting language to implement automatic activities and evaluate decisions on what transition to use.
- File type detection for converter use.

This will provide a Java-based CDR that match the current CDR, but the webforms, QA and converters will still be loosely linked with XML-RPC. If need be, we could put this system into production, but to get an improved system we must:

5.2.2. Make all converters local converters

There are two conversion systems in CDR. A local one operating on non-XML files. Based on the file type it simply calls subprograms to convert to HTML. The other one for XML files is a separate website written in Java, and it should be integrated into CDR for the sake of efficiency.

5.2.3. Better webforms

Integrate webforms into CDR and use AJAX. Perhaps use eXist XUpdate to change values in an XML file. The AJAX will make the webforms work faster as many things wont require a page reload.

5.2.4. Google Earth network link interface

Google Earth has the ability to discover feature maps on a compliant website. This is done by a web service script that lists the available maps and is called a Network Link. Google Earth reloads the page every time it needs to, so new maps can be added or removed dynamically. It requires some modifications to the conversion service. It must e.g. be possible to find all XML schemas having a conversion to KML and the ability to add a bounding box as a parameter to the conversion.

The feature is a simpler version of the Inspire interface, and can serve as a stepping stone to the Inspire interface.

5.2.5. Inspire interface

Some of the deliveries on CDR are stations that contain lat/long coordinates in typically decimal degrees format. We convert these to Google Earth KML with a simple stylesheet, and the benefit is that it is possible to verify the location. We could do the same for Inspire. First task would be to make CDR discoverable as a geoportal according to the OGC Webservices specification for XML files with lat/long information. Typically, the file contains the station data. The other data in the table would be available as attributes.

The methods are mandatory for CDR to expose a WFS (Web Feature Service):

- GetCapabilities
- DescribeFeatureType
- GetFeature

The GetCapabilities call will return a list of feature types. In CDR's case, typical feature types would be OzoneStation and EperFacility.

Then the client will ask: 'What is an OzoneStation' with the DescribeFeatureType method. CDR will respond that an OzoneStation has the following properties: StationName:string, TypeOfStationO3:string, TypeOfStationEoI:string etc. These properties are easy to enumerate if the table is defined in DD.

The client calls GetFeature. Inside GetFeature are up to several Queries. The Queries will be performed and the results in GML will be concatenated to a result set and sent to the client. The client can ask that only a subset of the available properties be returned.

All of this is so complicated that it requires a conversion service embedded in CDR.

The second task would be to make regular GIS files (GML, ESRI) available.

*

The end result will be a CDR with all the above features implemented as a Java Web application, that can be deployed on Unix, Linux or MS-Windows by member countries.

5.3. Implementing SEIS

SEIS is about getting access to and sharing data and information that is handled in information systems and in organisations all over Europe, both at local, intraregional, national as well as the European level. Its guiding principles are:

- information is managed as close as possible to its source
- information is provided once and shared with other for many purposes
- information should be readily accessible to end-users

We have discussed what effect SEIS would have on Reportnet with regards to:

a) How to find the datasets amongst all the other documents on the Internet
b) How to describe what the dataset contains so it is possible to decide before download whether it is relevant or not.
c) How to ensure it is not obsolete. E.g. a newer edition is available
d) How to understand what format the dataset is in, and
e) How to convert the dataset to the needed format.

... and we drew up five scenarios that progressively loosened up the requirements for the providers. Before we go into the scenarios, we need to establish the following definitions:

1. A *delivery envelope* is a container that holds the *set of files* that constitutes the delivery. The metadata is on the envelope; not the individual files.

2. If a delivery/file is tagged with the triple: ROD dataflow, country, reporting period, it is easy to handle, as it constitutes an answer to a reporting obligation. But the downside is that many datasets from NGOs don't have obligations in ROD.
3. QA reports - including the automatic - are the data requester's comment on the deliveries. Therefore it shouldn't be the providers' task to develop automatic QA scripts for their own reports.
4. The CR mentioned below is an object oriented search engine described in section 5.3.6.

5.3.1. Scenario 1:

Scenario 1 is intended to be a narrow extension of the existing Reportnet system. It is just an expansion with distributed repositories. In short bullets:

- ✔ Every country gets a national CDR called NDR
- ✔ The system operates with delivery envelopes
- ✔ The NDRs make metadata available in RDF form or as a Webservice
- ✔ Only ROD dataflows and they follow reporting guidelines

The way we would solve it is to get CR to harvest all national CDRs. Then we can search on dataflow, period and coverage and get a list of relevant datasets.

The scripts to do automatic QA would be located centrally at the requester. The QA service would run the QA scripts on the files at each NDR and will store the results back to the NDR as feedback reports in the envelopes. Same as today.

5.3.2. Scenario 2:

- ✔ *Every country has a static website* that holds the envelopes. This means we can't expect the features in CDR to be available, such as a QA system, posting of feedback reports in the envelope etc.
- ✔ They make metadata available in RDF form or as a Webservice
- ✔ Only ROD dataflows and they follow reporting guidelines
- ✔ Still envelopes

In this scenario we can still do a dataflow search: In CR you can make a dataflow search that will show the files in the envelope. With the metadata of the files (URL, XML schema) we can implement a conversion service in CR, if the user wants to see the data as e.g. MS-Excel.

Similarly, the automatic QA can be implemented because it only needs the files' URL and schema.

What about the QA feedback reports – both manual and scripted? We would put them on a central location (e.g. the ETC website). The QA report has feedback and the URL of the file that it is about. We can then have CR harvest the QA website. When a user then makes a dataflow search on CR, and finds a delivery he's interested in, CR will show that this delivery has a QA report posted on another site.

The problem of redelivering the files and having old feedback reports is solved by putting timestamps on files and checking at frequent intervals for changes of files.

5.3.3. Scenario 3:

- ✔ Every country has a static website
- ✔ They make the metadata available in RDF/Webservice
- ✔ *Not only ROD dataflows*
- ✔ Still envelopes

How can we find invasive species data when we can’t rely on an obligation id?

The metadata for the envelope will contain a list of files belonging to the envelope. As in scenario 2, the automatic and manual QA will be posted on a central website, and CR will establish the link between the envelope and the QA reports on its factsheet for the envelope.

But the big question in scenario 3 is; how do we find the information, when there is only limited metadata? Unlike Google, the systems can’t get information from the content of the files, as datasets tend to be only numbers, and Google’s page ranking mechanism won’t help much either. There are few links to these datasets.

To help, we must most likely use some Web 2.0 techniques and something called social bookmarking made famous by Del.icio.us. Visitors to CR, who has found some envelopes they are interested in, would be able bookmark the URLs with a personal tag. The tag entry field can have suggestions taken from GEMET. The visitor tags *will be seen by other visitors* to CR, and thereby they *amend* the metadata already attached to the envelope. In principle it is a simpler form of QA. The tagging can be made easy to do for the users by offering a couple of simple mechanisms: A plugin to install in IE and Firefox ála. del.icio.us, a bookmarklet or encourage websites with environmentally relevant content to add a little static HTML on every page that leads to the tagging form.

Usecase: Dataproviders will tag the records at the source by tags that describe the intended publishing purpose. For example, an Article 17 dataflow data provider will mark a species factsheet with “article 17, species, habitats, EU directive”, whilst a user will search for the actual species name and “migration patterns” and not find the Article 17 factsheets that contain information of species that happen to have migrated North. So, aside from suppliers tagging their own deliveries, we could let users that search on CR, tag pages with their preferred labels, which will complement the description of the record. If by chance an envelope has been tagged with a word in GEMET, we can use the GEMET hierarchy to do guided navigation. Since GEMET is translated, if someone has tagged an envelope with ‘haruldane liik’ it can also be found under ‘rare species’.

Other ideas to consider:

Listmania: you are looking on the URL factsheet on CR and if this URL is on another user’s list of bookmarks you can look at that person’s whole list.

Users’ descriptions: the previous users of a dataset can post comments about it on CR (or QAW) saying things like “comprehensive”, “obsolete”, “covers 1999 - 2005” and all of the above together.

Popularity comments can say something about the quality of the dataset expressed as "*X out of Y users thought this dataset was useful*". We don't recommend outright ranking because people might rank a dataset by the user's intended purpose and not as an overall ranking of the dataset. An example would be that a user interested in invasive species would assess a given dataset as "1", whereas a user interested in migratory patterns would assess the same dataset as "5". However, we *can* construct a popularity ranking out of indicators such as how many times the dataset occurs on a bookmark list.

Users who produced something from datasets they found on CR, could make a special tag that is a declaration: "*We [Eurostat] produced the information at URL X from this source*". They would put the same "tag" on all the sources they used. They would help themselves because they could be notified of updates to the sources, and it would help the community.

Search inside previous search. If a user has searched on 'species' he can then try out out a more narrow by using clues from the titles and descriptions provided in the metadata of the hits from the first search.

Fuzzy search: If a user has typed 'alein species' into the search field, and some envelopes have been tagged with 'alien species', it could make a spell check using the list on known tags, and like Google suggest 'Alien species'.

5.3.4. Scenario 4:

This scenario is set up to match a situation, where EEA can make some requirements to the information provider, but without requiring a total reorganisation of an existing website. One such example would be Norway's www.environment.no.

- ✔ Every country has a static website
- ✔ They make metadata available for CR
- ✔ Not only ROD deliveries
- ✔ *No envelope concept*
- ✔ *Metadata is on the published file*
- ✔ *Files can be published in multiple formats*

How do we avoid the problem that the user only finds and downloads one file out of a dataset of several? We use the Qualified Dublin Core "relation" element:

- use "references" to point to station data on the ETC website
- use "requires" to point to other files in the same delivery
- use "replaces" for old versions, e.g. previous deliveries

If you publish a file in multiple equal formats, then you must use "isFormatOf" and point to the original format. If this is not available, then point to a non-existent or unique URL. Conceptually, alternative formats of a file can be seen as a owner provided conversion. On CR's factsheets of a file, they will show up on the label "isFormatOf" If EEA has found conversions at its own Conversion service, then these could also show up at the same place.

All of the metadata discussion from scenario 3 also applies.

5.3.5. Scenario 5:

This scenario is relevant if we expect to find and use information produced by NGOs and individuals, who has no incentive nor means to implement the requirements made in the previous scenario.

- ✔ Every SEIS participant has a static website
- ✔ Any type of dataset
- ✔ No envelopes concept
- ✔ *Not just countries – also NGOs and municipalities*
- ✔ *No published metadata*

These websites are unharvestable by CR. We don't have the same resources as Google, and if the metadata isn't made available, then it will cost too much to scan it. In addition, the previously discussed issue of having no textual content in the dataset still apply. We must assume that CR knows nothing about the data until someone adds metadata. Therefore;

- Users find the file with Google or plain surfing the web
- We suggest a plugin like the del.icio.us that stores the tags at EEA
- The "relation" links discussed in scenario 4 would be maintained centrally by volunteers/ visitors à la. Wikipedia
- The coverage will represent a special spatial problem, as data might not neatly cover country boundaries.
- The plugin could have a feature to look up in CR the URL your browser is showing. It would be called "What does CR know about this page?" and would show user-supplied descriptions, relations, QA reports, etc.
- As an alternative to the plugin del.icio.us also has a bookmarklet written in Javascript.
- In addition CR could be a search engine in the "Instant Search Box" in IE7 and Firefox, but this does not replace the plugin or the bookmarklet.

In our opinion, it will be very difficult to achieve critical mass in this scenario, unless a significant amount of datasets are available under the other scenarios. We have already done similar tasks for years, but in a uncoordinated way, and with limited success. For example, since 2003, EEA has had a website called SERIS. On user suggestion, links to national reports have been tagged with metadata; country code, title etc. and then put into a database, and there is a webpage to show these reports at http://www.eionet.europa.eu/seris. This is not the only site we have. By centralising the metadata to one database, but keeping the frontend, we could achieve a more integrated system.

If we implement the Web 2.0 ideas, we could call it *Reportnet 2.0*.

5.3.6. CR description

It is easiest to describe what CR is by comparing it with Google. If you let Google index CDR, it would discover a set of webpages, MS-Excel, MS-Access and shapefiles. This is because Google is not object oriented. It only deals with the content, and not the container. CR on the other hand would see deliveries, and files belonging to those deliveries. It requires metadata that goes beyond Dublin Core. The metadata must describe not just the files themselves, but the relationship between files.

There doesn't have to be a physical object either. Content Registry also indexes concepts as objects. Let's make an example. We don't harvest datasets from member countries, so to follow the example you must mentally substitute a dataset with a Reportnet Delivery.

To find all Reportnet deliveries available from Latvia, run this query:

http://cr.eionet.europa.eu/search_results.jsp?pred_TYPE=Reportnet+Delivery&pred_DC%3ACOVERAGE=Latvia&search_precision=exact&SearchType=SEARCH

You should get about 125 hits. Clicking on one of the **»** sends you to the metadata page. If the object had been an XML file, we would have indexed the file name, type, size, XML Schema identifier as well as more user supplied metadata describing what it contains. But as soon as we have the XML schema or DTD identifier, CR could use the Reportnet conversion service to provide more convenient formats.

5.3.7. Scenario 4 example – how to link Norway's data to SEIS

To make the concept more clear we offer an example of the requirements Reportnet would impose on a member country in scenario 4 – which is judged to be closest to a full SEIS system.

Johnny Auestad gave us a pointer to http://www.environment.no/. If you go to this website, you will see it is full of data on Norway. How do we make this data available for SEIS? Discoverability is the key.

While the Norwegian website is pretty easy to navigate, if you want data for all of Europe, you have to look at 30+ websites with different navigational structure. So much time wasted that you'll never get to do the actual work.

For the example, we'll use the Contaminated soil data. Somewhere in the middle, there is a Contaminated soil link. Click on it. Then click on Contaminated soil on the right side under Maps and data. At the bottom of this page there is a See meta-data (in Norwegian) As you can see, the Norwegians have done a great job filling out the meta-data, and there is even a link to EEA's Reporting obligation listed as: Rapporteringsforpliktelse. Unfortunately, all the metadata is not easy to find in Google, as it is shown for a *human* audience.

What we need is a machine-readable description of the meta-data. CR uses a format called RDF. It could look like this:

```
<seis:Dataset rdf:about="http://www.miljostatus.no/datakat/18548">
    <dc:title>Miljøgifter : Forurenset grunn</dc:title>
    <dc:date>2006-05-14T08:49:29Z</dc:date>
    <dc:language>en</dc:language>
    <seis:country>NO</seis:country>
    <dc:coverage>1989-2007</dc:coverage>
...
    <rod:obligation rdf:resource="http://rod.eionet.eu.int/obligations/33" />
</seis:Dataset>

<seis:File rdf:about="http://www.miljostatus.no/datakat/18548?xls">
```

```
 <dc:identifier
rdf:resource=”http://62.92.43.137/InternetReportServer?...Format=Excel”/>
 <dc:format>application/ms-excel</dc:format>
 <dcterms:isFormatOf rdf:resource="http://www.miljostatus.no/datakat/18548"/>
</seis:File>

<seis:File rdf:about=”http://www.miljostatus.no/datakat/18548?csv”>
 <dc:identifier
rdf:resource=”http://62.92.43.137/InternetReportServer?...Format=CSV”/>
 <dc:format>text/plain</dc:format>
 <dcterms:isFormatOf rdf:resource="http://www.miljostatus.no/datakat/18548"/>
</seis:File>

<seis:File rdf:about=”http://www.miljostatus.no/datakat/18548?xml”>
 <dc:identifier
rdf:resource=”http://62.92.43.137/InternetReportServer?...Format=XML”/>
 <dc:format>text/xml</dc:format>
 <dcterms:isFormatOf rdf:resource="http://www.miljostatus.no/datakat/18548"/>
</seis:File>
```

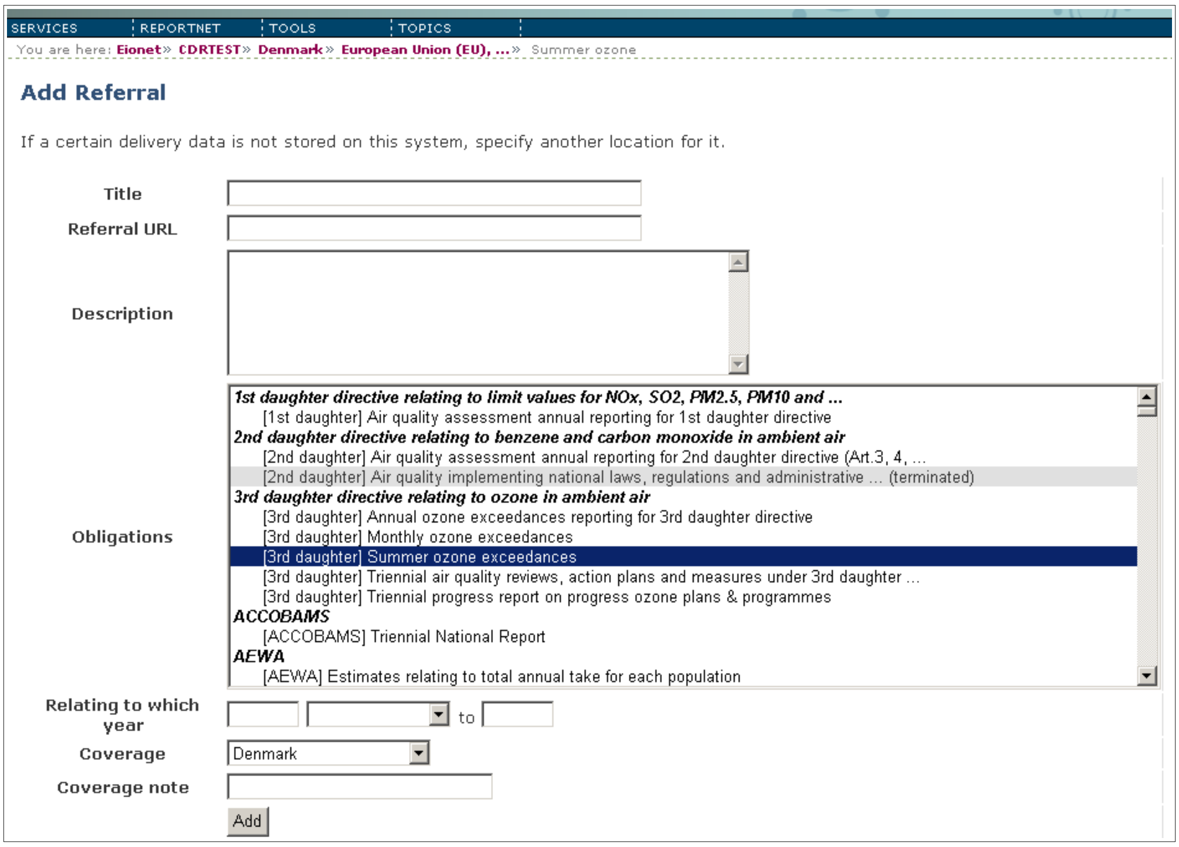
The magic that makes all this work is that the XML tags are standardised. The <dc:title> will always be used for the title of the document. And you can make your own tags if you coordinate it with the stakeholders. Therefore the <rod:obligation> tag will used by everyone who needs to relate a dataset to an obligation in ROD. If you load the above snippet into CR, you will then be able to ask which datasets are reporting on the ROD obligation *Contaminated* soil identified by the globally unique ID *http://rod.eionet.eu.int/obligations/33*. You can then use the other tags to filter even more.

When you have found the dataset you can follow the <dcterms:isFormatOf> tag in CR to discover that the dataset is available in three formats. Excel, CSV and XML. You can then download the one you can use best.

All Norway has to do is to make a file available on the website containing the metadata for all her datasets. We’ll let CR grab the file at intervals, and now Norway’s datasets can be found.

5.3.8. Scenario 5 example

All this about asking the requesters to add metadata may sound complex, but Reportnet already does it. There is a mechanism on CDR, where the data provider can add the necessary metadata to a file stored on his own system.



You are here: Eionet» CDRTEST» Denmark» European Union (EU), ...» Summer ozone

Add Referral

If a certain delivery data is not stored on this system, specify another location for it.

Title	
Referral URL	
Description	
Obligations	***1st daughter directive relating to limit values for NOx, SO2, PM2.5, PM10 and ...*** [1st daughter] Air quality assessment annual reporting for 1st daughter directive ***2nd daughter directive relating to benzene and carbon monoxide in ambient air*** [2nd daughter] Air quality assessment annual reporting for 2nd daughter directive (Art.3, 4, ... [2nd daughter] Air quality implementing national laws, regulations and administrative ... (terminated) ***3rd daughter directive relating to ozone in ambient air*** [3rd daughter] Annual ozone exceedances reporting for 3rd daughter directive [3rd daughter] Monthly ozone exceedances [3rd daughter] Summer ozone exceedances [3rd daughter] Triennial air quality reviews, action plans and measures under 3rd daughter ... [3rd daughter] Triennial progress report on progress ozone plans & programmes ***ACCOBAMS*** [ACCOBAMS] Triennial National Report ***AEWA*** [AEWA] Estimates relating to total annual take for each population
Relating to which year	to
Coverage	Denmark
Coverage note	
	Add

Illustration 1: Filling out metadata for a referral to a SEIS dataset

As you can see, the metadata being asked for is prearranged with the requesters. Obviously on CDR it can only be used for deliveries related to an obligation and for users at national focal points. The challenge is to make something similar for the other communities.

5.3.9. Conclusion: What is needed to implement SEIS?

The previous paragraphs described how Reportnet could evolve to accommodate the SEIS design of distributed systems. In our opinion, SEIS in its final form will look a lot like scenario 5. The main question is; which scenario do we target for the first iteration – the first limited system? The big work is in scenario 3, and if we only go for scenario 2, we implement a subset of the proposals.

To implement scenario 3 we need:

- Improvements to CR.
    - Since it was built for dataflow searches, it will need a database more suited for large data amounts – such as e.g. Nutch.
    - User interface modifications
- A system to hold user-supplied metadata such as tags, comments and QA reports. The system must (like del.icio.us) have a notion of user workareas, for a user to be able to correct/delete his own bookmarks and other metadata. It can be standalone, or it can be a component of CR or QAW.

– A system to run QA scripts on a file when it is discovered or a notification has been sent that the file is released. This system already exists in Reportnet, but must be modified to post the QA reports at a location other than the repository of the file the QA was run on.

5.4. Improvements to the QA system

The QA system is implemented as a webservice. CDR calls the webservice with the name of the file to check, the web service downloads the file, and runs the QA tests. CDR then picks up the result a minute or more later. We were forced to do the implementation this way, because CDR is written in Python, and there was no XQuery module available for that language. And to some extent SEIS has made that necessity a virtue. With distributed data repositories, we need to specify mechanisms that makes it possible for national repositories to call the central QA system to get the QA run both on-demand and as a batch process.

One major limitation we found when implementing the QA for Art. 17 was that our original assumption that QA scripts would do relatively simple checks, didn't hold true. There was a need to check values against very large lookup tables, such as all sites in Europe. The problem is a matter of efficiency. The lookup table is stored as a simple XML file on a remote system, it is loaded in by the XQuery script, and then parsed in memory. Only then is it used to check a value in the delivery. Since Art. 17 consisted of hundreds of XML factsheets, the lookup file would be downloaded and parsed hundreds of times.

To make it more efficient, and to be able to use even bigger lookup tables – such as the list of all known species in Europe – it is necessary to parse the lookup table and store it in an XML database that supports XQuery directly and the database must be part on the QA system.

5.5. Improving Data Dictionary

5.5.1. On making DD support XML schemas

DataDictionary has had less success than expected. The main problem as I see it, is that DD is organised around relational tables. These are often not convenient to deliver data in. For instance, for Article 17, we have three XML file formats, and if we should have used DD to describe the file format, we would have ended up with more than 60 tables. One solution to this is to extend DD to become a full XML schema designer, but we think it is not reasonable, because:

- Given the resources we have, we can never compete with such powerful XML Schema design tools like XMLSpy, Stylus Studio or Oxygen. These are commercial software. But there are also very nice freeware tools like XmlPad or XmlFox. There is just no point in creating a new toolkit in DD when there already exist very good toolkits.
- It should not be important which tool is used to create the schema, as long as the schema and possibly its elements and types become registered in DD.
- Looking at the practices of US EPA (http://iaspub.epa.gov/emg/xmlsearch$.startup) and other institutions listed at http://www.xml.gov/registries.asp, they too are not providing any toolkits for creating the schemas (at least I couldn't find any). Instead

they focus on registering available schemas which, naturally, is important in data exchange networks.

- It is hard to imagine that the current data definers in EEA and ETCs will start creating XML Schemas in DD. Even if the UI would be completely graphical, it is vital that they still have knowledge about XML Schema. For example US EPA has recognized that it takes 3 levels of expertise in order to develop an XML Schema for a data flow:
    - Business area experts with programmatic interest in the data
    - Data systems experts (like experts on CDR, Data service, DEMs, etc.)
    - Experts familiar with XML Schema design, XML technologies and standards

However, extending DD with an XML Schema *registry* is a justified idea. It could serve as a starter and growing field for more complex features to come in the feature. For example DD could become intelligent enough to parse registered schemas, extract elements and types from there and store them in the relation data model. It would then be possible to provide search functions across such elements and types. Or there might be query languages to search directly in the schema files. In any case, simple features like

- registering an XML Schema,
- providing the schema's URL or uploading the schema file into DD database (useful for parties who don't want to maintain the schema at their own site);
- specifying its description, status and other metadata;
- associating it with a certain data flow;
- keeping history per schema or dataflow;

would make DD much more useful.

5.6. The QA workbench (QAW)

When the data reporter releases an envelope, it is *the* delivery from that country. Frozen in time. But there can be many reasons, why the requester can't use it directly. The files can be zipped, in an inconvenient format or split over several files. The problem is that the requester is not and should not be allowed to fix issues like this directly on CDR, because it compromises the integrity of the delivery. Then, how can we help the requester?

What the requester does today is to collect the deliveries to a directory on his own PC. He "cleans" the data and then typically imports it into a database or spreadsheet. There are a small number of typical operations that the requester does to check the quality and aggregate the data. It is those operations we want to make easier.

The QA workbench (QAW) will be an option the requester can use for a while and then continue with his own tools when the QAW is insufficient.

These are the main features of the workbench.

- The system is a website, where each user has a personal work space. Alternatively there are project spaces for particular dataflows. Users can create folder structures as they like, copy-paste across folders will be possible.

- The system handles both XML and non-XML data.
- It will be possible to document the steps to take (the methodology) in a sort of README file for the dataflow.

Getting the data to work on:

- For repositories that follow SEIS scenarios 1 and 2, the QA-worker can query CR via QAW on the following parameters: country, obligation id and reporting period. He will then get a list of envelopes of which he can choose to import some or all to the QAW. He can delete envelopes already imported and let the import override existing envelopes on the QAW.
    - The import includes envelope metadata, all files, automatic QA-information, and the events log.
- For other repositories, the QA-worker can query CR via QAW on other metadata, find file to import or he can type the URL of the file and get QAW to retrieve it
    - In both cases, QAW will query CR to get available metadata and QA reports on the file
    - When a file is imported from a repository, en entry in an event log will be written stating the file's provenance. (Whence it came)
- When a file is uploaded to QAW from the user's harddisk, the system will ask how it was produced, and where it came from.

Analysing and correcting the data:

- Having imported a set of envelopes, he now proceeds to clean up the data. This could be:
    - Expand zipped files
    - Download individual files, convert or edit them, and then reupload
    - Add supporting files
- As part of his evaluation of a file the user will be able to add tags, QA comments and other metadata to the file. Some of these he can make public, and they will be added to CR's knowledge of the file. Here we have a potential repository for the manual QA reports discussed in Scenario 2.
- The QAW will have some scripting languages that enables the QA worker to write scripts that works on the data. The scripts will also be stored on the QAW in a script repository with metadata making it possible for the user to find and use them again when the next reporting period comes along.
- For MS-Excel files, it could be made possible to write a script to extract a given cell or column in a given sheet from a collection of MS-Excel files and then store the extraction in a database system.
- For text files, it could be made possible to write scripts in AWK using the JAWK library. Like for XQuery scripts, the AWK scripts would be stored on QAW.

- For XML files, the QAW must support XSLT and XQuery scripts.
- It will be possible to run the QA scripts that the QA system has stored for the automatic QA. One reason is that the QA-worker might have fixed some issues himself, and then wants to rerun the QA on that file.

Converting and aggregating data:

At some point after correction of the data the individual country contributions must be aggregated to one dataset. Most often the desired result is to put the data into a relational database.

- It must be possible to select a group of XML files in the work space, and then run the conversion on all in a sequence.
- The user can make some of each file's metadata available to the converter as parameters. For instance, the country code and reporting period, if they have been set on all the files being used in the conversion.
- If the output from the conversion is in a specific table-oriented XML format, it will be possible to save the output into a database attached to the QAW.
- This database system embedded in the QAW can be MySQL or Hsqldb. With Hsqldb, the database will look like a file in the user's work area, that can be downloaded, copied and deleted. With MySQL it could be possible to connect from the user's pc with a client-side tool or ODBC.

6. References

GML Application Schema
http://en.wikipedia.org/wiki/Geography_Markup_Language#Application_schema
MySQL
http://www.mysql.com/

HSQLDB
http://hsqldb.org/

Shape Viewer
http://www.qarah.com/shapeviewer/

Alfresco Intelligent File System
http://www.alfresco.com/
http://www.alfresco.com/products/aifs/

eXist
http://exist.sourceforge.net/

JA-SIG CAS
http://www.ja-sig.org/products/cas/ A system for authentication

JAWK: AWK written in Java
http://sourceforge.net/projects/jawk

RDF Primer
http://www.w3.org/TR/rdf-primer/