B.I.R.O

Best Information through Regional Outcomes

A Public Health Project funded by the European Commission, DG-SANCO 2005

WP 11: Web Portal

Document v1.0

April 2009

NOKLUS

Norwegian Quality Improvement of Primary Care Laboratories

Boks 6165
5892 Bergen- NORWAY
Contact person
Svein Skeie
Tel: +47 51519828
E-mail: svskeie@online.no

Table of contents

1. Introduction

The BIRO project has developed and demonstrated a model with strategic, technical, statistical and IT solutions that enable the building of safe multi-national shared information systems for sensitive clinical data. It has also done all the work necessary to implement the model for diabetes. The model, however, is generic and shall be made freely available. The web portal shall present both the generic mechanisms and the diabetes specific application. The web portal shall describe the structure of the project, the work and products of each work package and the partners in the BIRO consortium. This much could be handled with a standard web application having menu navigation to texts, diagrams and documents.
In the BIRO model information and results from an operational shared information system are disseminated via its web portal, largely in the form of displays of indicators. There will be many indicators and data will be collected on an on going basis. Without automation of updates the task of keeping information and results current will be costly and tedious. Integration with the Report-templates, data dictionaries and clinical knowledge base is required to provide transparency of what the results are based on. These requirements for automation, integration and transparency are important for the efficient operation of an active BIRO type shared information system. It is therefore not sufficient that the web portal shall just document the project and provide static examples of indicator presentations. The web portal must also be a custom programmed operative component in the BIRO model and a generic version of it must be made available as an open source, easily adaptable, software component in the BIRO Transfer of Technology collection.

The duality of purpose of the web portal complicates the issue of “who is the target audience” and thereby the content and the design of the user interface. Work package 7 (Report templates) made recommendations based on the role such a web portal would have as part of a sustainable shared information system. It described distinct provisions for audiences of governance, health care and research, and people with some reason to seek information on diabetes. However, although the BIRO project has built all the functionality of a shared information system it has been a limited project. It does not have a great deal of data and collection will cease on completion of the project. Its primary focus has been on solutions, mechanisms and enablement. The most important products of the project are the generic BIRO model, the application of that to diabetes and of making these solutions freely available to other longer term projects that will establish and run sustainable information systems. The main value of the indicators presented here lies not in the limited information they provide on the state of diabetes and diabetes care in Europe but in the functionality they provide for others to use and what they illustrate of mechanisms and pertinent reports generated by semi-automated data collection, statistical processing and display. Thus, for this web portal it is people who have a vested interest in the establishment of sustainable multi-national shared information systems who are the main audience targeted. Texts have though been included that attempt to describe the project to any curious reader.
For subsequent projects like EUBIROD that will use the BIRO model to establish a sustainable shared information system the focus will be on the data

they collect and the analyses they produce. Consequently they will have other audiences and other agendas. Each such project will make their own choices of how they will relate to their audience and what services they will provide. BIRO will not decide this for them in advance, so in its dual roles as a description of the BIRO project and as an operative component designed for transfer of technology the technical design of this web portal will be confined to providing core BIRO model functionality plus mechanisms for managing general content of texts, documents and menus.

2. Objectives

• To present the work and achievements of the completed BIRO project.
• To provide and demonstrate mechanisms for informative and effective dissemination of results generated by a sustainable shared information system based on the BIRO model.
• To provide an open source software component for transfer of its technology at low cost.
• To provide a single point of access to the main products developed by the BIRO consortium.

3. Materials and methods

If the BIRO website was to be just a single installation that presented and demonstrated the BIRO project software tools would have been chosen solely on considerations of technical excellence and convenience for us as developers and those who will support it. But the web application has a second function. It implements behaviour that is important to the indicator displays and the BIRO model. It contains custom programmed mechanisms and it interacts with other BIRO applications and software. It is an integral part of the BIRO model as it will be implemented in each sustainable shared information system. As such a version of it must be made available as a component in the BIRO Transfer of Technology software collection. Otherwise every organisation that chose to implement the BIRO model would have to start from scratch and develop web applications having equivalent mechanisms and functionality. Therefore the web portal is required to comply with the policy of BIRO Transfer of Technology. Thresholds for implementing applications shall be as low as possible, both in terms of work involved and cost. Not only shall the application, source code, and documentation be made freely available but recipients shall not be required to purchase any commercial proprietary software in order to implement, adapt, or further develop. Because of this only open source tools and resources were acceptable for building and implementing the web portal and to the selection criteria was added the requirement that they be readily available and very widely used and supported.

For the implementation the following technologies were chosen:

Apache HTTP Server
Apache is a free and open source HTTP server application maintained under the auspices of the Apache Software Foundation. It is run mainly on Unix and Unix-like operating systems but is available also for a variety of others, including Windows and Mac OS X. Apache can serve both static and dynamic pages on the web.
As of March 2009 Apache served over 46% of all websites and over 66% of the million busiest. (http://news.netcraft.com/archives/2009/03/15/march_2009_web_server_survey)

MySql or PostgreSQL (relational database management system)
Of the serious free open source RDMS systems available MySql is the most used worldwide and PostgreSQL claims to be the most advanced. Both provide all the functionality required for the web portal application and a lot more. BIRO has used PostgreSQL for its other database applications, but the issue here is a separate web server installation so that is not a decisive factor. We have used both and the application runs unchanged on either.

Drupal (content management framework)
Drupal is an open source content management framework. It is widely used and runs on both MySQL and PostgreSQL. It has many modules available, there is a lot of development going on and it is very easy to install. It consists of a small core, and gives the developer a comprehensive interface for implementing custom made modules. The framework gives us content-management, security, database-connectivity and menu-system. It also gives

the ability for non-technical people to maintain and update the portal without any need for coding. (http://www.drupal.org)

PHP
PHP is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It can be deployed on most web servers and on almost every operating system and platform free of charge. PHP is installed on more than 20 million websites and 1 million web servers.
Having chosen Apache and Drupal PHP was a clear choice.

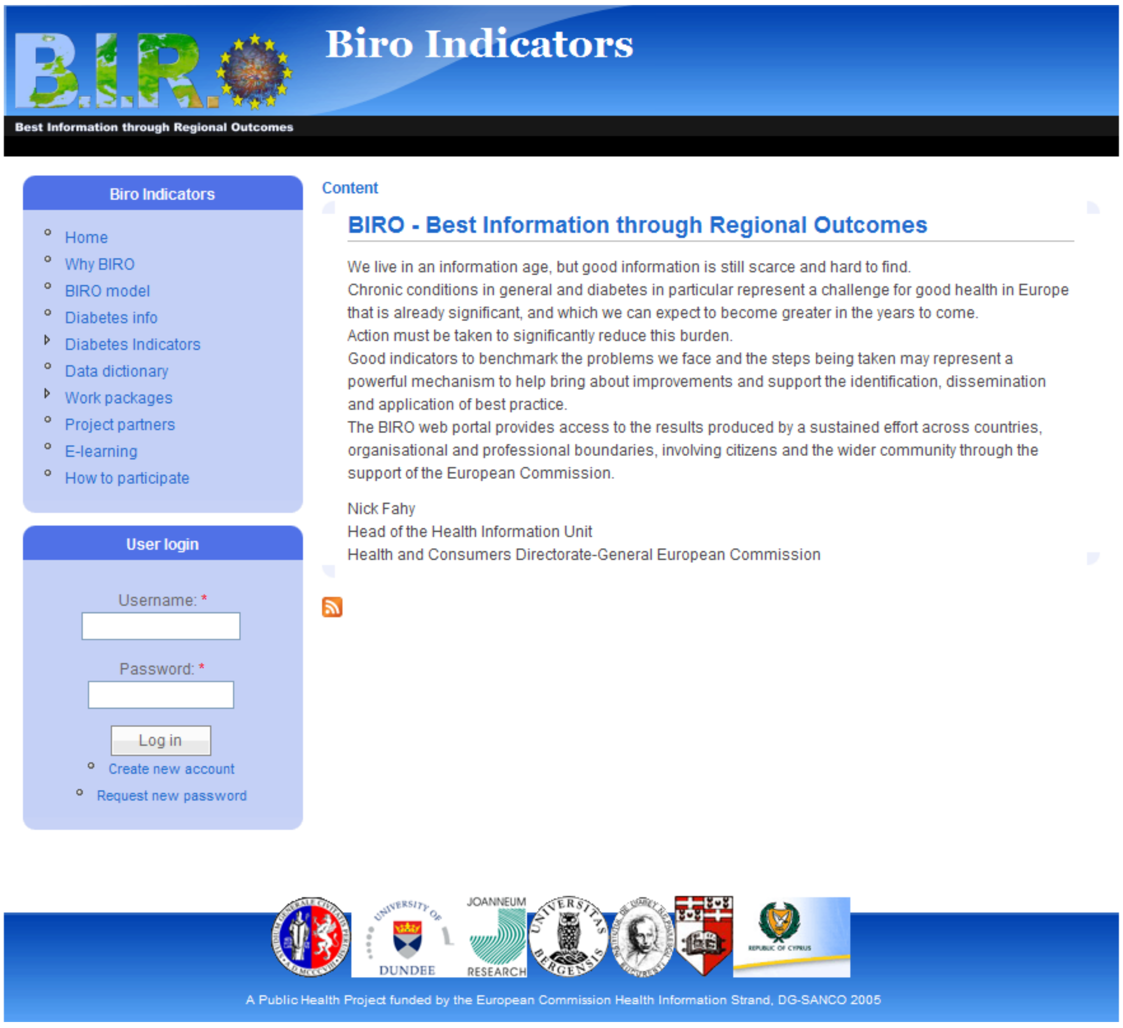
Should some future user of the system already have other databases, content management frameworks, programming languages etc. that they prefer to use they are of course at liberty to do so. The documentation and source code provided will describe mechanisms, essential functionality and how the application interacts with other BIRO software. This will minimise the effort required to port and adapt to other technologies.

4. Results

Static content
Static content and mechanisms for management of texts, documents and menus.

The work of building the web portal has had two phases. The first was building the basic web application with hierarchal menu navigation to the static texts and documents that provide information about the BIRO project. The Drupal content management framework was the primary tool used to do this. It has facilities and utilities for forming the appearance and layout of the web pages, security, database-connectivity, managing static menus, inserting texts and linking to documents etc. These same Drupal facilities will enable people with only quite modest web building skills to easily modify these aspects of the application to their own requirements and maintain them. Drupal's own documentation explains how to do this. (http://www.drupal.org)

Menu:
The menu is the primary navigation mechanism. It is always present on the screen. There are relatively few items initially shown but it is hierarchal where required and allows the user to drill down through layers of categories to the topic sought.


Biro Indicators

Best Information through Regional Outcomes

Biro Indicators
- Home
- Why BIRO
- BIRO model
- Diabetes info
- Diabetes Indicators
- Data dictionary
- Work packages
- Project partners
- E-learning
- How to participate

User login

Username: *

Password: *

Log in
- Create new account
- Request new password

Content

BIRO - Best Information through Regional Outcomes

We live in an information age, but good information is still scarce and hard to find.
Chronic conditions in general and diabetes in particular represent a challenge for good health in Europe that is already significant, and which we can expect to become greater in the years to come.
Action must be taken to significantly reduce this burden.
Good indicators to benchmark the problems we face and the steps being taken may represent a powerful mechanism to help bring about improvements and support the identification, dissemination and application of best practice.
The BIRO web portal provides access to the results produced by a sustained effort across countries, organisational and professional boundaries, involving citizens and the wider community through the support of the European Commission.

Nick Fahy
Head of the Health Information Unit
Health and Consumers Directorate-General European Commission

A Public Health Project funded by the European Commission Health Information Strand, DG-SANCO 2005

Figure 1: Home page

Menu items

The choice of primary menu items and the static material they accessed were decided by consensus amongst the partners in the consortium. The menu items in the top layer are:

◦ Home
◦ Why BIRO
◦ BIRO model
◦ Diabetes info
▹ Diabetes Indicators
◦ Data dictionary
▹ Work packages
◦ Project partners
◦ How to participate

Where a menu item is linked to a single document the task of providing the document was allocated to a single person or group. These items were:

- A text for the **Home** page text to capture interest and endorse the project: (Fabrizio Carinci)
- "**Why BIRO**". A common language text to explain the relevance of the BIRO project. (Peter Taverner)
- "**BIRO model**". A common language description of the BIRO model that explains the key problems encountered when building an international shared information system for sensitive clinical data, and the solutions BIRO developed to overcome them. (Peter Taverner)
- "**Diabetes info**". A concise description of what diabetes is and its impact. (Svein Skeie)

For the menu item "**Work packages**" each partner was required to provide a short abstract describing the work and results of their work package plus a document of the full deliverable for their work package. When the "Work package" menu item is selected it first drops down a list of the work packages in the project. When a 'work package' is selected from this the abstract is displayed. From this the reader can click a link that will fetch and display the full 'Deliverable' document for that work package.

For the menu item "**Project Partners**" all partners were asked to provide a short document with relevant information about them selves. This item shows first a list of the partners and from there the document provided by the partner selected.

The item "**How to participate**" does not invite participation in the BIRO project. That project is complete. Instead it shows some information about the follow up project EUBIROD and provides a link to its web portal. EUBIROD will use the BIRO model and the work done by BIRO in applying the model to diabetes to establish a sustainable European Union shared information system for diabetes and will be interested in recruiting new partners to fill out its coverage of Europe. Also those who wish to establish their own shared information system using the BIRO model will be able to get help there. The partners of the BIRO consortium are also partners in the EUBIROD project. Considering the generic nature and wide applicability of the BIRO model it would though make good sense to maintain a general and independent interface and service for this via the BIRO web portal.

The menu items "**Diabetes indicators**" and "**Data dictionary**" are not static. They are programmatically generated by the custom programmed modules for display of indicators and the browsing of metadata. Those modules build their menus from data supplied by applications external to the web portal application.

Indicator presentation.

Design goals:
The results for each indicator will be presented on the web site using a collection of tables, charts and texts that are specific to each. The issue here is what qualities are required of the manner of doing this:

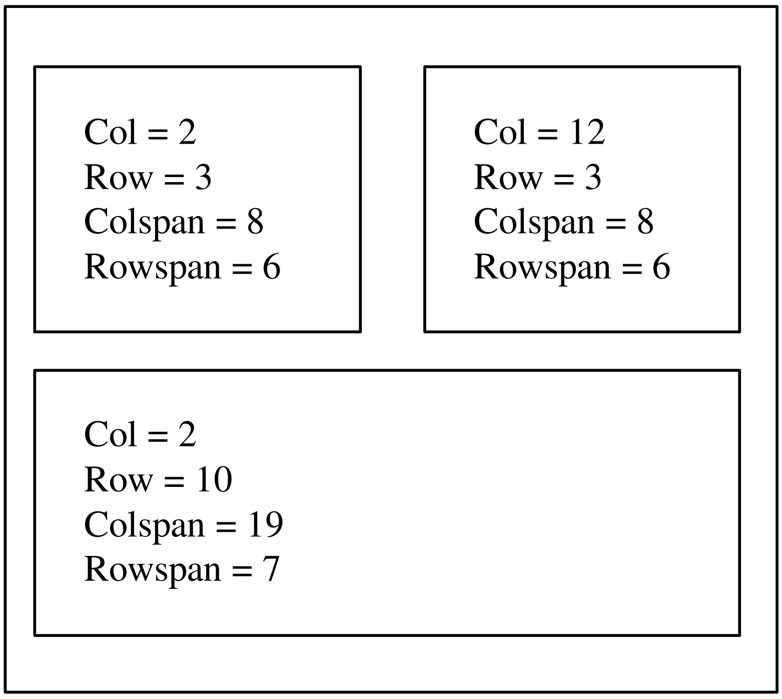
Automation: With shared information systems of the type the BIRO model was designed for data will be collected on an on going basis. Analysis of these data and updating of the many indicators presented on the web will be done frequently. Unless automated this job will be tedious and costly.

The ability to establish and maintain the choice of indicators and how they will be presented without web programmer intervention: It is work done in Clinical-review, Reports-template and Statistical-engine that is responsible for the clinical, epidemiological and graphical quality of the indicators. A solution that allows the work done by those specialists to be expressed directly on the web without requiring re-programming would simplify the task and reduce the cost of setting up and maintaining the indicators displayed. Being able to easily and cheaply make extensive adaptations is also a major factor in the suitability of the solution as a "Transfer of technology" component.

The custom Drupal module developed for Indicator presentation:
This module provides two services:

- A dynamic menu that provides the reader with an overview of the indicators that can be viewed and the ability to browse or step through them. (Figure 6) The menu is hierarchal and organised by the same theme chapters and order as expressed in the Report Templates register. The module programmatically generates this menu based on the data provided from the Reports template. When changes are made to the Report Templates the process to generate the menus must be re-run, but this is easy and quick.
- A page on which the indicator selected from the menu is displayed. The key to this part is that the module concerns itself solely with display of data supplied. What it will display, for a given indicator, is supplied as a collection of "display ready elements". These elements can be of type text, table or chart/diagram. Specification of where on the page each element shall be shown and how big it shall be is also supplied as data. For positioning and sizing of elements the module operates with a simple grid. This makes it easy for the person designing the page layouts for the elements. For each element the position of the top left corner is given as a column and row coordinate, the width as a number of columns and the depth as number of rows.

Figure 2 Example of element layout.



When a new indicator is to be added to those already supported:

- It is named and the justification for its choice is documented with supporting references (Clinical review).
- The calculation is defined and the list of involved variables listed. (Clinical review)
- The strata and form of presentation is defined (what type of chart, table or diagram etc). (Report templates)
- A program is written that will instruct the Statistical Engine how to access the primary data, perform the data processing and statistical analysis required, and generate the output required as 'display-ready' elements. (Charts, tables, diagrams)
- Any other 'display-ready' elements required are created. For example: blocks of explanatory text. (Report templates)
- A file name is registered, for each display element, for where the content of that element will be stored. (Report templates)
- The layout of the elements that will make up the presentation of the indicator on the web page is specified. (Report templates)

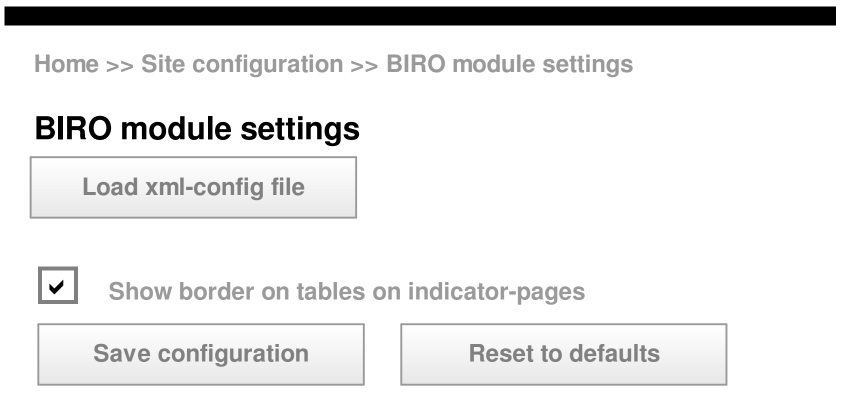
All of this will be done in the general data system for the information system and the data will be stored in its database.

When the collection of indicator base data has been changed the web portal "Indicator presentation module" needs to be re-configured. To do this, the layout specifications for each display element are collected from the Report templates register and delivered to the web application. The module's configuration process is then invoked. This will generate a new menu and prepare the module for actual display. The module is now capable of showing each indicator presentation. For each indicator it has a list of all the elements in the display. For each element it has the pathname of the file containing the content of the element and the specifications for its position and size on the page. This configuration process need be repeated only when the data it depends on from the Report templates register are changed.

Updating the results presented on the web can be done whenever and as often as desired simply by running the Statistical Engine programs. The definitions of the indicator and of what to produce as output are built into the program. One program is run for each indicator and each program will generate one or more display elements (tables or charts). Each element generated is written to its specified file, which is where the web application will read from when it needs it. Processing the data can be done whenever it is appropriate because of new data or defined time intervals etc. and that which is displayed on the web will thus be automatically updated. The web application is not in any way involved, it just displays what is in the files without knowing when or how it got there.

Technical details

When this module is installed it will create two working tables and generate an administration screen for managing the configuration process (Figure 3.).



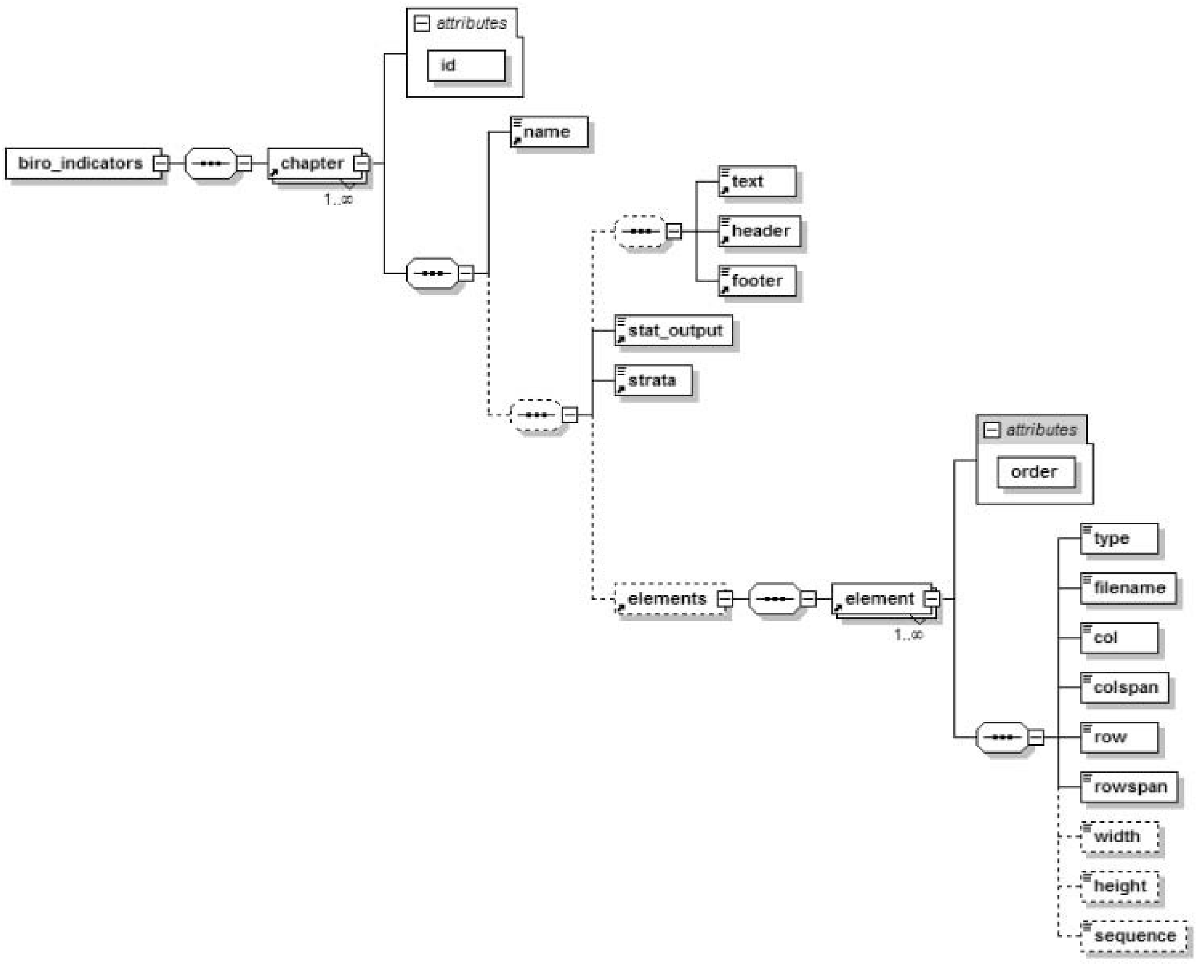
✓ Show border on tables on indicator-pages

Figure 3. Configuration management screen

The configuration specifications for indicators are held in xml blocks in the Reports Template register, one for each indicator. When specifications are added or changed the configuration process for the module must be run. The first step is to collect all the indicator xml blocks into a single xml document and make it available to the web application. A schema has been defined for this “indicator configuration data document” (Figure 4).

Currently building the xml data document and uploading it to the server is done manually.
The remaining steps are done programmatically and are invoked via the configuration management screen provided by the module. The xml data document is validated against the schema. If valid, the module will traverse the xml and update the two tables in the database created for this. The indicator menus will then be rebuilt based on the new data. If the xml is not valid an error will be displayed and the configuration will not be changed.

Figure 4 Schema defining the indicator configuration data document



In the xml configuration block the content of each element is specified as the pathname of a file. The content of each file is “display ready” and requires only to be positioned and sized. Files for text and table elements contain html. Image elements contain a picture. It is this feature that enables programmatic updating of indicator displays as part of the statistical
engine ‘refresh with new data’ process. The statistical engine generates output for the web as ‘display ready’ tables and charts and writes them to the designated location using filenames specified in the xml configuration blocks. With each reprocessing of the data the content of these files is replaced. The web indicator module simply displays what is currently there. The output to be generated by the statistical engine for each indicator is specified as part of the Report Templates.
To define the layout of the indicators the following information is required:

Name	Usage	Description
Chapter ID	Mandatory	The chapter
Name	Mandatory	The name of the indicator
Text	Optional	Descriptive text
Header	Optional	Header-text for indicator
Footer	Optional	Footer-text for indicaor
Statistical	Optional	Type of output (histogram, line, etc)
Strata	Optional	
Sortorder	Mandatory	The order in which the indicators will show in menu

For each element specified for an indicator the following information is required:

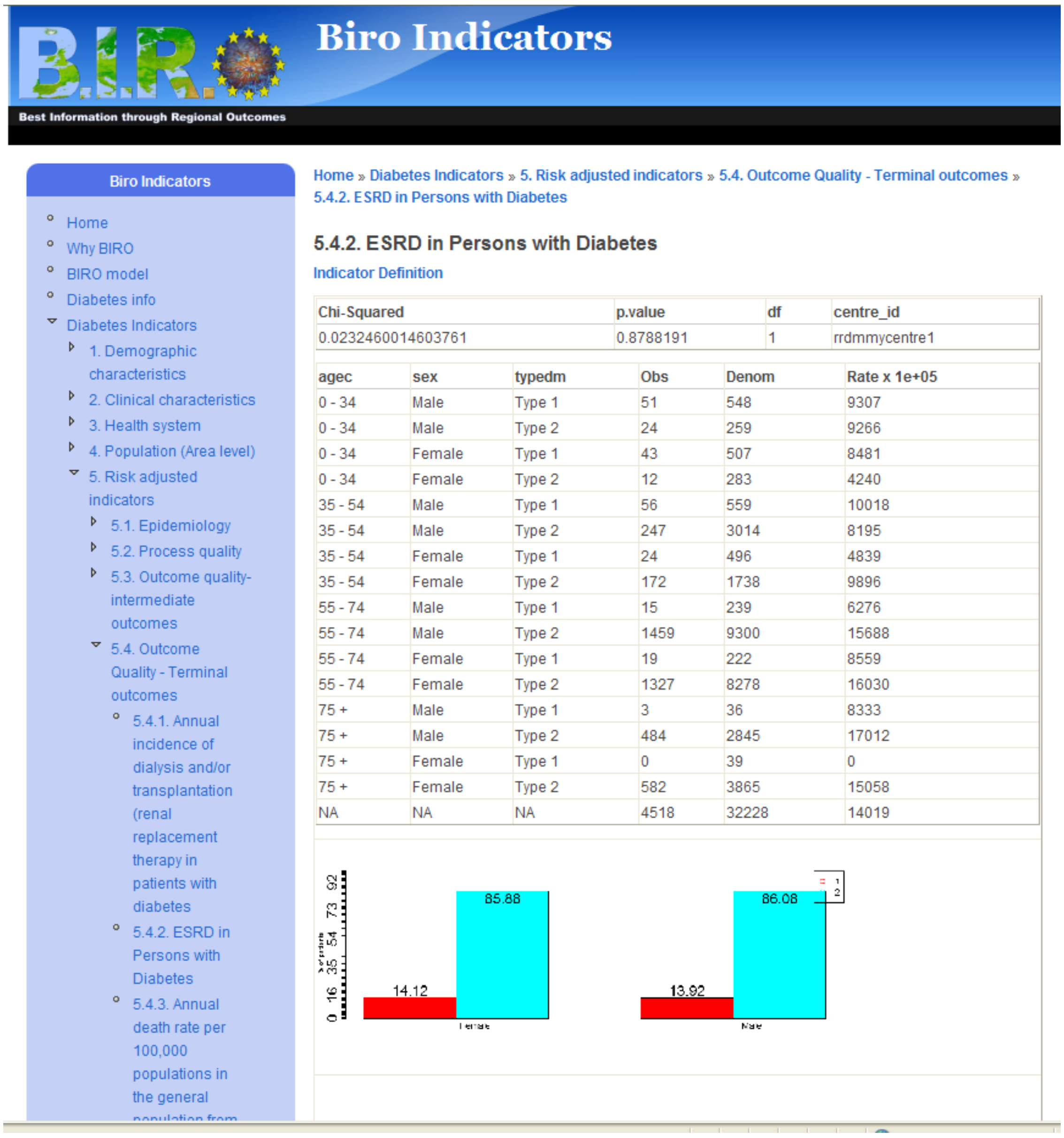
Name	Usage	Description
Chapter ID	Mandatory	The chapter ID. References the chapterid in the corresponding indicator
Type	Mandatory	image, text or table
Filename	Mandatory	The name of the file containing the data.
Row	Mandatory	
Column	Mandatory	
Rowspan	Optional	Default 1
Column-span	Optional	Default 1
Vieworder	Mandatory	Display-order of element.
Width	Optional	Applies only to images.
Height	Optional	Applies only to images.
Sequence	Optional	If type is table, this attribute can be used to identify which table to pick from file, if the file contains more than one table.

Figure 5. An example of an xml configuration file

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Sample XML file generated by XMLSpy v2008 sp1 (http://www.altova.com)-->
<biro_indicators xsi:noNamespaceSchemaLocation="indicators.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <chapter id="1">
            <name>Demographic characteristics</name>
      </chapter>
      <chapter id="1.1">
            <name>Age (Classes)</name>
            <text/><header/><footer/>
            <stat_output>histogram</stat_output>
            <strata>Gender</strata>
            <elements>
                  <element order="1">
                        <type>image</type>
                        <filename>1_1.png</filename>
                        <col>1</col>
                        <colspan>1</colspan>
                        <row>1</row>
                        <rowspan>1</rowspan>
                        <width>50%</width>
                        <height>50%</height>
                  </element>
                  <element order="2">
                        <type>table</type>
                        <filename>1_1.html</filename>
                        <col>1</col>
                        <colspan>1</colspan>
                        <row>2</row>
                        <rowspan>1</rowspan>
                  </element>
                  <element order="3">
                        <type>text</type>
                        <filename>desription1_1.html</filename>
                        <col>1</col>
                        <colspan>1</colspan>
                        <row>3</row>
                        <rowspan>1</rowspan>
                  </element>
            </elements></chapter>
      <chapter id="1.2">
            <name>Gender</name>
            <text/><header/><footer/>
            <stat_output>histogram</stat_output>
            <strata>Age</strata>
            <elements>
                  <element order="1">
                        <type>image</type>
                        <filename>1_2.png</filename>
                        <col>1</col>
                        <colspan>1</colspan>
                        <row>1</row>
                        <rowspan>1</rowspan>
                        <width>50%</width>
                        <height>50%</height>
                  </element>
                  <element order="2">
                        <type>table</type>
                        <filename>1_2.html</filename>
                        <col>1</col>
                        <colspan>1</colspan>
                        <row>2</row>
                        <rowspan>1</rowspan>
                        <sequence>2</sequence>
                  </element>
            elements>
      </chapter>
</biro_indicators
```

All the menu items in the Indicator menu link to the indicator module. The section of the indicator data to display is given to the module as a parameter. For example: “http://<hostname>/?q=biro/1/1”, (where “biro” is the name of the Drupal indicator module), points to the module with the indicator “1.1” as a parameter. The module queries the biro_indicators table for the specified indicator. If found it then queries the biro_data table for the elements to show. It generates the html code needed for laying out the elements according to the settings for each element and returns this to the Drupal engine. The indicator menus (Figure 6) are rebuilt each time the configuration process is run.

Figure 6: Example of indicator display accessed via menu selection. For each indicator display the number of elements used is not restricted. That which does not fit on the visible page can be made visible by scrolling. In the example shown there are two more charts below that which is shown on the screen.



Biro Indicators

- Home
- Why BIRO
- BIRO model
- Diabetes info
- Diabetes Indicators
  - 1. Demographic characteristics
  - 2. Clinical characteristics
  - 3. Health system
  - 4. Population (Area level)
  - 5. Risk adjusted indicators
    - 5.1. Epidemiology
    - 5.2. Process quality
    - 5.3. Outcome quality- intermediate outcomes
    - 5.4. Outcome Quality - Terminal outcomes
      - 5.4.1. Annual incidence of dialysis and/or transplantation (renal replacement therapy in patients with diabetes
      - 5.4.2. ESRD in Persons with Diabetes
      - 5.4.3. Annual death rate per 100,000 populations in the general

Home » Diabetes Indicators » 5. Risk adjusted indicators » 5.4. Outcome Quality - Terminal outcomes » 5.4.2. ESRD in Persons with Diabetes

5.4.2. ESRD in Persons with Diabetes

Indicator Definition

Chi-Squared	p.value	df	centre_id
0.0232460014603761	0.8788191	1	rrdmmycentre1

agec	sex	typedm	Obs	Denom	Rate x 1e+05
0 - 34	Male	Type 1	51	548	9307
0 - 34	Male	Type 2	24	259	9266
0 - 34	Female	Type 1	43	507	8481
0 - 34	Female	Type 2	12	283	4240
35 - 54	Male	Type 1	56	559	10018
35 - 54	Male	Type 2	247	3014	8195
35 - 54	Female	Type 1	24	496	4839
35 - 54	Female	Type 2	172	1738	9896
55 - 74	Male	Type 1	15	239	6276
55 - 74	Male	Type 2	1459	9300	15688
55 - 74	Female	Type 1	19	222	8559
55 - 74	Female	Type 2	1327	8278	16030
75 +	Male	Type 1	3	36	8333
75 +	Male	Type 2	484	2845	17012
75 +	Female	Type 1	0	39	0
75 +	Female	Type 2	582	3865	15058
NA	NA	NA	4518	32228	14019

Documentation and source code for this module can be viewed in appendix 1

Browsers for data dictionaries

Transparency is a vital aspect of quality assurance of systems that process data and present results. A design goal of the web portal was to make provision for this. Some of the relevant information will be in documents that explain the essential features and mechanisms of the system or provide background information on the partner's data sources. Much of the essential data will be, or should be, in the form of 'dictionaries'. For example: definitions of variables and indicators, documentation from each partner of their compliance with the standard variable definitions etc. Data of this type should not be buried in large documents. Such dictionaries should, at the very least, be made browser able. 'Dictionary browsing' however makes just one category of information available at a time and each item within it must be searched for. In some contexts, like viewing and evaluating indicators, a single click by the user should suffice to bring into view a collection of those metadata most relevant to enhance the understanding of that indicator and to enable evaluation of the quality of the underlying data. A custom Drupal module has been developed which provides a mechanism both for 'stand alone' browsing of data dictionaries accessed via the menu and a simple form of bundling metadata with indicators so that for each indicator its definition metadata and the definitions of the variables involved are linked to and can be directly accessed from the display of the indicator.

Custom Drupal module for browsing data dictionaries

This module provides a generic mechanism for programmatically generating a browser within the web application from a structured file of data. Intervention from a web programmer is not required. The menu generated and the data displayed on selection of a menu item are determined entirely by the data file provided. This offers an easy way of providing access to dictionaries of base data. It is particularly useful where the data are labile. The data can be maintained in some application external to the web and the module will programmatically update the browser on demand. This ability to generate 'within web portal browsers' without programmer intervention also contributes significantly to the adaptability of the web portal and enhances its value as a 'transfer of technology' component.

The solution implemented in the web portal provides browsers for two dictionaries; (i) the common variable set and (ii) the definitions of the indicators. Included in the in the indicator metadata is a list of the variables involved and from this the reader can drill down to their definitions by clicking on a variable. The display will then jump to that variable definition displayed in the variable dictionary. For greater reader convenience these metadata are also accessible directly from the displays of indicator results. When viewing an indicator result the reader can click a button on that page which will cause a jump to the Indicator definitions.

The solution for "context bundling of metadata" implemented on the BIRO web portal is somewhat restricted and rudimentary but it is a good start to an important function. Subsequent projects, like EUBIROD, can be expected to develop enhanced versions that allow access to a greater range of metadata and improve the convenience and presentation of the metadata associated with each indicator display.

Technical details

The module requires that a manually initiated configuration process is run initially and whenever changes in the underlying data require it. For this the module currently requires that data for the dictionaries are provided in xml documents uploaded to a designated area on the web server. The configuration process will then use these to populate its relational tables in the web-server database and will then generate the menu required for browsing. The module provides an administration screen for this.
For each dictionary for which a browser is generated the configuration process will add a name to a list that the reader can choose from. These will be shown when the reader selects the menu item "Data dictionary".

Database tables for the Data dictionary module:

biro_datadict

Name	Datatype	Key	Description
Reference	varchar(255)	PK	Variable (Ex.BIRO001)
Field_name	varchar(255)		Variablename (Ex. PAT_ID)
Parameter	varchar(255)		Description of variable (Ex. PatientID)
Datatype	varchar(255)		

biro_datadict_enum

Name	Datatype	Key	Description
Reference	varchar(255)	PK1	Variable (Ex. BIRO001)
Enum_code	Int	PK2	Ex. 1
Value	varchar(255)		Value of code (Ex Type 1)

biro_crossref

Name	Datatype	Key	Description
Chapter	varchar(255)	PK	Ex 1.1
Target	varchar(255)		Reference to indicator
Name	Text		Ex Age (Classes)

biro_crossref_stratum

Name	Datatype	Key	Description
Chapter	varchar(20)	PK1	Ex 1.1
Stratum	varchar(40)	PK2	Reference to variable, Ex DOB

biro_crossref_output

Name	Datatype	Key	Description
Chapter	varchar(20)	PK1	Ex. 1.1
Output	varchar(20)	PK2	Output-type, ex Histogram

Documentation and source code for this module can be viewed in appendix 2

5. Optimising for Transfer of Technology

To make the application suitable as an Open Source component in the BIRO "transfer of technology" collection there are two major considerations. One is to use exclusively Open Source applications and resources to build and implement it. The other is to make adaptability a strong feature. Make it easy for others to modify a copy of the application to suit their needs while still retaining the core functionality that is essential to the BIRO model.
While static content can readily be changed using the Web Content Management framework the core BIRO functionality lies in the custom programmed modules.
Both custom modules have been designed so that what they display is determined by data that has been generated and supplied to them by applications external to the web application. With the display of indicators for example, not only is the choice of which indicators to display and the content and layout of each presentation determined completely by data supplied from outside the web, but because display elements are supplied in "display ready" form also things like the quality of chart and diagram images etc. are determined outside the web application and no re-programming of the web modules is required. What these custom modules show and express can thus be easily and radically changed without programming. Only when new or enhanced functionality is required will there be a need for programming. When that happens it would be of great value if the resulting enhancements or variations were also made available as Open Source software.

6. Summary

The Web Portal developed in WP11:

- Describes the BIRO project and the BIRO model.
- Demonstrates the functioning BIRO model as applied to diabetes
- Has developed a mechanism for informative, low overhead presentation of frequently updated results.
- Has developed a mechanism for generating browser interfaces for data dictionaries.
- Is generic, easily adaptable and technically suitable as a 'Transfer of technology' component.

As such WP11 has achieved its objectives.

The BIRO project is to be followed by the EUBIROD project which will establish a sustainable shared information system for diabetes in Europe. EUBIROD will use both the generic BIRO model and the extensive work done in BIRO on applying that model to diabetes. But while EUBIROD has been enabled by BIRO it is a distinct project with its own objectives. Its focus will be on the data it collects and analyses and what uses are made of them. As such it will create its own web portal and can do this by adapting and enhancing the BIRO web portal component. The BIRO web portal will be retained to present the work and achievements of the BIRO project.

The ultimate goal of the BIRO initiative has been the establishment of a sustainable shared diabetes information system for the European Union. It is the tightly related follow up project EUBIROD that shall realise the final stage of

this. In the process software from BIRO will be significantly enhanced and added to. Thus EUBIROD will complete BIRO also in another respect. By the time EUBIROD is fully functional the software made available via BIRO Transfer of Technology should be much improved, thoroughly tested, and there should be much more of it. This will include significant enhancements to the web portal component which then will be an example of a single purpose tested and functional website for a sustainable shared information system.
Some examples of things that will affect existing or require new software are:

- The structure of data repositories in BIRO have been strongly influenced by the work having been done in parallel and largely autonomous work packages. In EUBIROD these should be coalesced into a cohesive and more functional data base.
- Considerable development of indicator presentation and other report forms is expected. To handle this substantial software enhancement is expected.
- Applications that simplify the work required to administer Report Templates and primary data dictionaries will probably be developed.
- Other functions, like "E-Learning" will be made available.
- It is possible that applications that can make the primary harvesting of data easier and more standardised could be developed. (After all, the most critical factor for success with an information system is obtaining enough good data).
- And more.

EUBIROD will directly apply the BIRO model and its specific application to diabetes, as could others who may wish to establish a shared information system for diabetes for some other region. However the principles of the model, its examples and the open source software it makes available can be applied equally well to other diseases. As such it would be advantageous to promote the BIRO model on a website where the model, the examples, the software collection and links to related projects are the primary focus rather than on one, like EUBIROD, where the focus will be on the data collected for a particular disease and the insights gleaned from them.

Making available useful generic methods and nurturing the concept and practice of Open Source software can reduce costs in the health care sector. It can make feasible things for which thresholds were formerly too high. It would help the proliferation of methods like BIRO, which in addition to their primary value as information systems would also promote acceptance of international standards and 'Best practice' methods.
All of which makes a strong case for providing support for a web portal that can:

- Accumulate links to other related projects,
- Accumulate or link to new or enhanced Open Source software,
- Promote and foster an Open Source community for projects and work of this type.

This could be done either by extending and supporting the BIRO web portal so that it can provide this service or by establishing a new web portal for that sole purpose.

The test installation of the BIRO web portal can be viewed at //BIRO.tysseit.no

Appendix 1

Documentation and source code for custom Indicator module

Info-file:

```
; $Id$
name = Biro Data
description = A block which shows the data in the BIRO data-tables.
core = 6.x
```

Install-file:

```
<?php

/**
 * Implementatin of hook_install
 *
 */
function biro_install(){
	drupal_install_schema('biro');
	db_query("insert into menu_custom values ('menu-biro','Biro Indicators','Biro
Indicators')");
}

/**
 * Implementation of hook_uninstall
 *
 */
function biro_uninstall(){
	db_query("delete from menu_custom where menu_name='menu-biro'");
	drupal_uninstall_schema('biro');
}

/**
 * Implementation of hook_schema
 *
 */
function biro_schema(){
	$schema['biro_indicators'] = array(
   'fields' => array(
      'header' => array('type' => 'varchar', 'length' => 255,'not null' => FALSE),
      'text' => array('type' => 'text', 'not null' => FALSE),
      'footer' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
      'chapter' => array('type' => 'varchar', 'length' => 255, 'not null' => TRUE),
      'indicator' => array('type' => 'text', 'not null' => FALSE),
      'stat_output' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
      'strata' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
      'sortorder' => array('type' => 'int', 'not null' => FALSE)),
   'primary key' => array('chapter'),
        );

        $schema['biro_data'] = array(
   'fields' => array(
       'chapter' => array('type' => 'varchar', 'length' => 255, 'not null' => TRUE),
       'type' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
       'filename' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
       'row' => array('type' => 'int', 'not null' => FALSE),
       'col' => array('type' => 'int', 'not null' => FALSE),
       'rowspan' => array('type' => 'int', 'not null' => FALSE),
       'colspan' => array('type' => 'int', 'not null' => FALSE),
       'view_order' => array('type' => 'int', 'not null' => TRUE),
       'width' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
       'height' => array('type' => 'varchar', 'length' => 255, 'not null' => FALSE),
                   'sequence' => array('type' => 'int', 'not null' => FALSE)),
   'primary key' => array('chapter', 'view_order'),
        );

        return $schema;
}
```

Module-file:

```
<?php
// $Id
/**
 * @file
 * Shows BIRO-indicators. The indicator to show is given as a variable, with chapter
name.
 *       Ex to show biro indicator 1.1.1 : ?p=biro/1.1.1
 */
/**
 * Implementation for hook_help
 *
 * @param unknown_type $path
 * @param unknown_type $arg
 * @return unknown
 */
function biro_help($path, $arg) {
        $output = '';
        switch ($path) {
                case "admin/help#biro":
                        $output = '<p>'.  t("Biro indicators.") .'</p>';
                        break;
        }
        return $output;
}

/**
 * Implementation for hook_perm
 *
 * @return unknown
 */
function biro_perm() {
	return array('access biro content');
}

/**
 * Implementation of hook_all
 *
 * @param string $chapter
 * @return string
 */
function biro_all($chapter) {

	$dir = "sites/default/files/biro/elements/";
	// Query for fetching the given indicator
	$indicator_query = db_query("SELECT * from biro_indicators where chapter =
'%s'", $chapter);


	// Did the query return any records?
	if ($indicator = db_fetch_object($indicator_query)) {
		// Put header and text at the top
		$page_content .= $indicator->header;
		$page_content .= $indicator->text;

		$tok = strtok($chapter, ".");
		$path = "/" . $tok;
		while ($tok !== false) {
			$path .= "/" . $tok;
			$tok = strtok(".");
		}

		$page_content .= '<p><a href="?q=datadictionary'.$path.'">Indicator
Definition</a></p>';

		// Query for fetching the elements in this indicator
		$data_query = db_query ( "SELECT * FROM biro_data where chapter =
'%s' order by view_order" , $chapter);

		$has_elements = false;

		// Loop on results from query on elements, and insert them in an array
		$elements_test = array();
		while ($data = db_fetch_object($data_query))			{
			$has_elements = true ;

			$elements_test[$data->row][$data->col]['pos'] = $data-
>row.'.'.$data->col;
			$elements_test[$data->row][$data->col]['type'] = $data->type ;
			$elements_test[$data->row][$data->col]['filename'] = $data-
>filename ;
```

```
				$elements_test[$data->row][$data->col]['colspan'] = $data-
>colspan ;
				$elements_test[$data->row][$data->col]['rowspan'] = $data-
>rowspan ;
				$elements_test[$data->row][$data->col]['orderview'] = $data-
>rowspan ;
				if ($data->type == 'image') {
					$elements_test[$data->row][$data->col]['width'] = $data-
>width ;
					$elements_test[$data->row][$data->col]['height'] = $data-
>height ;
				}
				$elements_test[$data->row][$data->col]['sequence'] = $data-
>sequence ;
			}


			// Did we find any elements for indicator?
			if ($has_elements) {

				if (variable_get('biro_border_width', 0)) {
					$page_content .= '<p><table border="2">' ;
				} else {
					$page_content .= '<p><table>' ;
				}

				foreach ($elements_test as $rows) {
					$page_content .= '<tr>';
					foreach ($rows as $cols) {
						if ($cols['type'] == "image") {
							$cell_content = '<img width="' .
$cols['width'] . '" height="' . $cols['height'] .'" src="'. $dir . $cols['filename'] . '" />';
						} else if ($cols['type'] == "table" || $cols['type'] ==
"text" ) {
							$tablefile = $dir. $cols['filename'];
							$fh = fopen($tablefile, 'r') ;
							$filedata = fread($fh, filesize($tablefile));
							fclose($fh);
							if ($cols['type'] == "table" &&
$cols['sequence'] != null) {
								$pos1 = 0;

								for ($i=0;
$i<$cols['sequence'];$i++){
									$pos1 =
stripos($filedata,'<TABLE',$pos1) + 1;
								}
								$pos2 =
stripos($filedata,'<TABLE',$pos1 + 6);

								if ($pos2 != ""){
```

```
                                                            $cell_content =
substr($filedata,$pos1-1,$pos2-$pos1);
                                                      } else {
                                                            $cell_content =
substr($filedata,$pos1-1);
                                                      }
                                                } else {
                                                      $cell_content = $filedata;
                                                }
                                          }
                                          $page_content .= '<td rowspan="'.
$cols['rowspan'] .'" colspan="' . $cols['colspan'] . '">' . $cell_content;
                                    }
                                    $page_content .= '</tr>';
                              }
                              $page_content .= '</table>' ;
                        }

                        $page_content .= $indicator->footer;
            }         else     {
                        $page_content .= 'No indicator found v1.0!';
            }
            return $page_content ;
}

function biro_menu() {
            $items = array();

            $items['menu-biro'] = array (
    'title' => 'BIROBIROBIRO',
    'description' => 'BIROBIROBIRO',
     'access arguments' => array('access biro content'),
      'type' => MENU_IS_ROOT
            );

            $items['admin/settings/biro'] = array (
                        'title' => 'BIRO module settings',
                        'description' => 'Description of your BIRO settings control',
                        'page callback' => 'drupal_get_form',
                        'page arguments' => array('biro_admin'),
                        'access arguments' => array('access administration pages'),
                        'type' => MENU_NORMAL_ITEM
            );

            $indicator_query = db_query("SELECT * from biro_indicators order by
sortorder");

            while ($data = db_fetch_object($indicator_query)) {

                        $tok = strtok($data->chapter, ".");
                        $path = "/" . $tok;
                        while ($tok !== false) {
```

```
                    $path .= "/" . $tok;
                    $tok = strtok(".");
            }

            $items["biro" . $path] = array(
       'title' => $data->chapter . ". " . substr($data->indicator,0,200),
                    'description' => $data->indicator,
                    'page callback' => 'biro_all',
                    'page arguments' => array($data->chapter),
                    'access arguments' => array('access biro content'),
                    'menu_name' => 'menu-biro',
             'weight' => $data->sortorder,
                    'type' => MENU_NORMAL_ITEM
            );
    }
    return $items;
}

function validate() {
    // Validate current configuration-file
    $dir = "sites/default/files/biro/";
    $xmlfile  = "biro_configuration.xml";
    $xmlschema = "biro_schema.xsd";

    $dom = new DOMDocument();
    //Load the xml document in the DOMDocument object
    $dom->Load($dir . $xmlfile);

    if (!$dom->schemaValidate($dir . $xmlschema)) {
            die ("$xmlfile is invalid!! Please correct file.\n");
    }
}

/**
 * Load configuration from xml-file
 *
 * @param unknown_type $form
 * @param unknown_type $form_state
 */
function load_configuration ($form, &$form_state)
{
    function sortorder($var) {
            $data = strrev($var);
            $fact = 1;
            $tok = strtok($data, ".");
            $sum = 0;
            while ($tok !== false) {
                    $sum += strrev($tok)  * $fact;
                    $tok = strtok(".");
                    $fact = $fact * 100;
            }
            return $sum;
```

```
}

function isnull($str)
{
        if (is_null($str)) {
                $ret = "NULL";
        } else {
                $ret = "'" . $str . "'";
        }
        return $ret;
}

function isvalnull($val)
{
        if (is_null($val)) {
                $ret = "NULL";
        } else {
                $ret = $val;
        }
        return $ret;
}

db_query("delete from biro_data");
db_query("delete from biro_indicators");

validate();

$dom = new DOMDocument();
$dom->load('sites/default/files/biro/biro_configuration.xml');

// Loop on all all chapters
foreach ($dom->getElementsByTagname('chapter') as $chapter)    {

        // Element?
        if ($chapter instanceof DOMElement) {
                // Find chapter-id
                $id = $chapter->getAttribute('id');

                // Prepare array
                $ch_array = array();
                $ch_array['elements_index'] = 0;
                $ch_array['chapter'] = $id;

                // Loop on all nodes in a chapter
                foreach (($chapter->childNodes) as $chapterNode) {

                        if ($chapterNode instanceof DOMElement) {

                                // if not an element...
                                if ($chapterNode->tagName <> 'elements') {

                                        // Add values to array
                                        switch($chapterNode->tagName) {
                                                case 'name': $ch_array['indicator'] =
$chapterNode->nodeValue; break;
                                                case 'text': $ch_array['text'] =
$chapterNode->nodeValue;break;
                                                case 'header': $ch_array['header'] =
$chapterNode->nodeValue;break;
                                                case 'footer': $ch_array['footer'] =
$chapterNode->nodeValue;break;
                                                case 'stat_output':
$ch_array['stat_output'] = $chapterNode->nodeValue;break;
                                                case 'strata': $ch_array['strata'] =
$chapterNode->nodeValue;break;
                                        }
                                } else {
                                        // Elements. Load them into an array
                                        foreach (($chapterNode-
>getElementsByTagName('element')) as $elements) {
                                                if ($elements instanceof
DOMElement) {
                                                        $element_array = array();
                                                        $element_array['vieworder']
= $elements->getAttribute('order');
                                                        $element_array['type'] =
$elements->getElementsByTagName ('type')->item(0)->nodeValue;
                                                        $element_array['filename'] =
$elements->getElementsByTagName ('filename')->item(0)->nodeValue;
                                                        $element_array['col'] =
$elements->getElementsByTagName ('col')->item(0)->nodeValue;
                                                        $element_array['colspan'] =
$elements->getElementsByTagName ('colspan')->item(0)->nodeValue;
                                                        $element_array['row'] =
$elements->getElementsByTagName ('row')->item(0)->nodeValue;
                                                        $element_array['rowspan'] =
$elements->getElementsByTagName ('rowspan')->item(0)->nodeValue;
                                                        $element_array['sequence']
= $elements->getElementsByTagName ('sequence')->item(0)->nodeValue;
                                                        if ($element_array['type'] ==
'image') {

        $element_array['width'] = $elements->getElementsByTagName ('width')-
>item(0)->nodeValue;

        $element_array['height'] = $elements->getElementsByTagName ('height')-
>item(0)->nodeValue;
                                                        }
                                                }
                                                $ch_array['elements_index'] += 1;

        $ch_array['elements'][$ch_array['elements_index']] = $element_array;
                                        }
                                }
                                }
                        }
                }

                // Ok, xml-file is read. Now generate sql-statements
                $query = "insert into biro_indicators
(chapter,header,text,footer,indicator,stat_output,strata,sortorder)values(" ;
                $query .= isnull($ch_array['chapter']) . ",";
                $query .= isnull($ch_array['header']) . ",";
                $query .= isnull($ch_array['text']) . ",";
                $query .= isnull($ch_array['footer']) . ",";
                $query .= isnull($ch_array['indicator']) . ",";
                $query .= isnull($ch_array['stat_output']) . ",";
                $query .= isnull($ch_array['strata']) . ",";
                $query .= sortorder($ch_array['chapter']) . ")";
                db_query($query);


                for ($idx = 1; $idx <= $ch_array['elements_index']; $idx += 1) {
                        $dataquery = "insert into biro_data (
chapter,type,filename,row,col,rowspan,colspan,width,height,view_order,sequence)
values (" ;
                        $dataquery .= isnull($ch_array['chapter']) . ",";
                        $dataquery .= isnull($ch_array['elements'][$idx]['type']) . ",";
                        $dataquery .= isnull($ch_array['elements'][$idx]['filename']) . ",";
                        $dataquery .= isvalnull($ch_array['elements'][$idx]['row']) . ",";
                        $dataquery .= isvalnull($ch_array['elements'][$idx]['col']) . ",";
                        $dataquery .= isvalnull($ch_array['elements'][$idx]['rowspan']) .
",";
                        $dataquery .= isvalnull($ch_array['elements'][$idx]['colspan']) .
",";
                        $dataquery .= isnull($ch_array['elements'][$idx]['width']) . ",";
                        $dataquery .= isnull($ch_array['elements'][$idx]['height']) . ",";
                        $dataquery .= isnull($ch_array['elements'][$idx]['vieworder']) .
",";
                        $dataquery .= isnull($ch_array['elements'][$idx]['sequence']) .
")";
                        db_query($dataquery);
                }
        }
}

/**
 * Implement hook_admin
 *
 * @return unknown
 */
function biro_admin() {

        $form['Load'] = array(
                '#type' => 'submit',
                '#value' => 'Load xml-config file',
                '#submit' => array('load_configuration'),
        );

        $form['biro_border_width'] = array(
   '#type' => 'checkbox',
   '#title' => t('Show border on tables on indicator-pages'),
   '#default_value' => variable_get('biro_border_width', 0),
        );
        //      $form['biro_border_width'] = array(
        //    '#type' => 'checkboxes',
        //    '#title' => t('Show borders in indicator-tables'),
        //    '#options' => array(t('Show border')),
        //         '#default_value' => variable_get('biro_border_width', array('TRUE')),
        //      );
        return system_settings_form($form);
}

?>
```

Appendix 2

Documentation and source code for custom dictionary browser module

Info-file:

```
; $Id$
name = Biro DataDictionary
description = A block which shows the Biro datadictionary
core = 6.x
```

Install-file:

```
<?php

/**
 * Implementation of hook_install
 *
 */
function datadictionary_install(){
	watchdog("DEBUG","datadictionary_install");
	drupal_install_schema('datadictionary');
}

/**
 * Implementation of hook_uninstall
 *
 */
function datadictionary_uninstall(){
	watchdog("DEBUG","datadictionary_uninstall");
	drupal_uninstall_schema('datadictionary');
}

/**
 * Implementation of hook_schema
 *
 */
function datadictionary_schema(){
	watchdog("DEBUG","datadictionary_schema");
	//datadictionary.xml
	$schema['biro_datadict'] = array(
		'fields' => array(
			'reference' => array('type' => 'varchar', 'length' => 255,'not null'
=> TRUE),
			'field_name' => array('type' => 'varchar', 'length' => 255, 'not
null' => TRUE),
			'parameter' => array('type' => 'varchar', 'length' => 255, 'not null'
=> TRUE),
			'datatype' => array('type' => 'varchar', 'length' => 255, 'not null'
=> TRUE),
			'enum_code' => array('type' => 'int', 'not null' => FALSE),
			'clinicaldefinition' => array('type' => 'text', 'not null' => FALSE),
                    'units' => array('type' => 'varchar', 'length' => 255, 'not null' =>
FALSE)),
                    'primary key' => array('reference'),
            );
        //datadictionary.xml
        $schema['biro_datadict_enum'] = array(
            'fields' => array(
                    'reference' => array('type' => 'varchar', 'length' => 255,'not null'
=> TRUE),
                    'enum_code' => array('type' => 'int', 'not null' => TRUE),
                    'value' => array('type' => 'varchar', 'length' => 255,'not null' =>
TRUE)),
                    'primary key' => array('reference', 'enum_code'),
            );
        //WP7CrossReference.xml
        $schema['biro_crossref'] = array(
            'fields' => array(
                    'chapter' => array('type' => 'varchar', 'length' => 255,'not null' =>
TRUE),
                    'target' => array('type' => 'varchar', 'length' => 255,'not null' =>
TRUE),
                    'name' => array('type' => 'text', 'not null' => FALSE),
                    'indicatortext' => array('type' => 'text', 'not null' => FALSE),
                    'numerator' => array('type' => 'varchar', 'length' => 255, 'not null'
=> FALSE),
                    'denominator' => array('type' => 'varchar', 'length' => 255, 'not
null' => FALSE),
                    'source' => array('type' => 'varchar', 'length' => 255, 'not null' =>
FALSE),
                    'algorithmcalculation' => array('type' => 'text', 'not null' =>
FALSE),
                    'algorithmoutput' => array('type' => 'text', 'not null' => FALSE)),
                    'primary key' => array('chapter'),
            );
        //WP7CrossReference.xml
        $schema['biro_crossref_stratum'] = array(
            'fields' => array(
                    'chapter' => array('type' => 'varchar', 'length' => 20,'not null' =>
TRUE),
                    'stratum' => array('type' => 'varchar', 'length' => 40,'not null' =>
TRUE)),
                    'primary key' => array('chapter', 'stratum'),
            );
        //WP7CrossReference.xml
        $schema['biro_crossref_output'] = array(
            'fields' => array(
                    'chapter' => array('type' => 'varchar', 'length' => 20,'not null' =>
TRUE),
                    'output' => array('type' => 'varchar', 'length' => 40,'not null' =>
TRUE)),
                    'primary key' => array('chapter', 'output'),
            );
```

```
	return $schema;
}
```

Module-file:

```
?php
// $Id
/**
 * @file
 */
/**
 * Implementation for hook_help
 *
 * @param unknown_type $path
 * @param unknown_type $arg
 * @return unknown
 */
function datadictionary_help($path, $arg) {
	watchdog("DEBUG","datadictionary_help");
	$output = '';
	switch ($path) {
		case "admin/help#datadictionary":
			$output = '<p>'.  t("Biro Datadictionary.") .'</p>';
			break;
	}
	return $output;
}

/**
 * Implementation for hook_perm
 *
 * @return unknown
 */
function datadictionary_perm() {
	watchdog("DEBUG","datadictionary_perm");
	return array('access biro content');
}



/**
 * Implementation of hook_all
 *
 * @param string $chapter
 * @return string
 */
function datadictionary_all($chapter) {

	$qry = db_query("select * from biro_crossref where chapter='$chapter'");
```

```
        $indicator_query = db_query("SELECT * from biro_crossref order by
chapter");

        $page_content = '<table border="1" valign="top">';
        $page_content .=
'<tr><td><em>Chapter</em></td><td><em>Indicator</em></td></tr>';

        while ($data = db_fetch_object($indicator_query)) {

                $tok = strtok($data->chapter, ".");
                $path = "/" . $tok;
                while ($tok !== false) {
                        $path .= "/" . $tok;
                        $tok = strtok(".");
                }
                if ($data->chapter == $chapter) {

                        $page_content .= '<tr bgcolor="lightgreen"><td><strong>'.$data-
>chapter.'</strong><td><strong>'.$data->name.':</strong><br>';
                        $page_content .= '<table border="2" valign="top"
bgcolor="lightyellow"><tr><th>Output<th>Stratum<th>Numerator<th>Denominator<
th>Source<th>Algorithm</tr>';
                        //$page_content .= '<tr><td width="20%">';
                        $page_content .= '<tr><td>';

                        $output_query = db_query("select output from
biro_crossref_output where chapter='$chapter'");
                        while ($output_data = db_fetch_object($output_query)) {
                                $page_content .= $output_data->output . '<br>';
                        }
                        $page_content .= '</td><td>';

                        $stratum_query = db_query("select stratum from
biro_crossref_stratum where chapter='$chapter'");
                        while ($stratum_data = db_fetch_object($stratum_query)) {
                                $page_content .= '<a
href="?q=datadictionary/variables/'.$stratum_data->stratum.'">' . $stratum_data-
>stratum . '</a><br>';
                        }

                        $page_content .= '</td><td>' . $data->numerator . '</td><td>' .
$data->denominator . '</td><td>' . $data->source;
                        $page_content .= '</td><td>' . $data->algorithmcalculation .
'</td>';
                        $page_content .= '</tr></table>';

                } else {
                        $page_content .= '<tr><td>' . $data->chapter
                                                        . '</td><td><a
href="?q=datadictionary'.$path.'">'.$data->name.'</a></td></tr>';
                }
        }
        $page_content .= '</table>';

        return $page_content ;
}

function datadictionary_variables($variable) {

        $page_content = '<p><table border="2"
valign="top"><tr><th>Reference<th>Name<th>Parameter<th>Datatype<th>Units</tr
>';

        $qry = db_query("select * from biro_datadict");
                while ($data = db_fetch_object($qry)) {
                        if ($data->field_name !== $variable) {
                                $page_content .=
                                        '<tr><td><a href="?q=datadictionary/variables/' .
$data->field_name . '">' . $data->reference . '</a></td>'.
                                                '<td><a href="?q=datadictionary/variables/'
. $data->field_name . '">' . $data->field_name .'</a></td>'.
                                                '<td><a href="?q=datadictionary/variables/'
. $data->field_name . '">' . $data->parameter . '</a></td>'.
                                                '<td><a href="?q=datadictionary/variables/'
. $data->field_name . '">' . $data->datatype . '</a></td>'.
                                                '<td><a href="?q=datadictionary/variables/'
. $data->field_name . '">' . $data->units . '</a></td>'.
                                        '</tr>';
                        } else {
                                $page_content .=
                                        '<tr bgcolor="lightgreen"><td><strong><a
href="?q=datadictionary/variables/' . $data->field_name . '">' . $data->reference .
'</a></strong></td>'.
                                                '<td><strong><a
href="?q=datadictionary/variables/' . $data->field_name . '">' . $data->field_name
.'</a></strong></td>'.
                                                '<td><strong><a
href="?q=datadictionary/variables/' . $data->field_name . '">' . $data->parameter .
'</a></strong></td>'.
                                                '<td><strong><a
href="?q=datadictionary/variables/' . $data->field_name . '">' . $data->datatype .
'</a></strong></td>'.
                                                '<td><strong><a
href="?q=datadictionary/variables/' . $data->field_name . '">' . $data->units .
'</a></strong></td>'.
                                        '</tr>';
                        }

                        if ($data->field_name == $variable and $data-
>datatype=="Enumerated") {
                                $enum_qry = db_query("select * from
biro_datadict_enum where reference='$data->reference' order by enum_code");
```

```
                                        $page_content .= '<tr bgcolor="lightgreen"><td><td
colspan="4"><table bgcolor="lightyellow"
border="2"><tr><th>Code<th>Value</tr><tr>';
                                        while ($enum_data = db_fetch_object($enum_qry)) {
                                                  $page_content .= "<tr><td>$enum_data-
>enum_code<td>$enum_data->value</tr>";
                                        }
                                        $page_content .= '</table></tr>';
                              }
                    }

                    $page_content .= '</table>';

          return $page_content;
}

function datadictionary_start() {
          $indicator_query = db_query("SELECT * from biro_crossref order by
chapter");
          $page_content .= '<p><table border="1">';
          $page_content .=
'<tr><td><em>Chapter</em></td><td><em>Indicator</em></td></tr>';
          while ($data = db_fetch_object($indicator_query)) {

                    $tok = strtok($data->chapter, ".");
                    $path = "/" . $tok;
                    while ($tok !== false) {
                              $path .= "/" . $tok;
                              $tok = strtok(".");
                    }
                    $page_content .= '<tr><td>' . $data->chapter . '</td><td><a
href="?q=datadictionary'.$path.'">'.$data->name.'</a></td></tr>';
          }
          $page_content .= '</table>';

          return $page_content ;
}

function datadictionary_menu() {
          $items = array();

          $items['admin/settings/datadictionary'] = array        (
                     'title' => 'BIRO datadictionary module settings',
                     'description' => 'Description of your BIRO settings control',
                     'page callback' => 'drupal_get_form',
                     'page arguments' => array('datadictionary_admin'),
                     'access arguments' => array('access administration pages'),
                     'type' => MENU_NORMAL_ITEM
           );

          $indicator_query = db_query("SELECT * from biro_indicators order by
sortorder");
```

```
while ($data = db_fetch_object($indicator_query)) {

        $tok = strtok($data->chapter, ".");
        $path = "/" . $tok;
        while ($tok !== false) {
                $path .= "/" . $tok;
                $tok = strtok(".");
        }

        $items["datadictionary" . $path] = array(
                'title' => "Indicators",
                'description' => $data->indicator,
                'page callback' => 'datadictionary_all',
                'page arguments' => array($data->chapter),
                'access arguments' => array('access biro content'),
                'menu_name' => 'menu-datadictionary',
           'weight' => $data->sortorder,
                'type' => MENU_NORMAL_ITEM
        );
}

$variable_query = db_query("select * from biro_datadict order by reference");
 while ($data = db_fetch_object($variable_query)) {
        $items["datadictionary/variables/".$data->field_name] = array(
                'title' => "Datadictionary",
                'page callback' => 'datadictionary_variables',
                'page arguments' => array($data->field_name),
                'access arguments' => array('access biro content'),
                'menu_name' => 'menu-datadictionary',
                'type' => MENU_NORMAL_ITEM
        );
}

$items["datadictionary/start"] = array(
        'title' => "Indicators",
        'page callback' => 'datadictionary_start',
        'access arguments' => array('access biro content'),
        'menu_name' => 'menu-datadictionary',
        'type' => MENU_NORMAL_ITEM
);

$items["datadictionary/variables"] = array(
        'title' => "Datadictionary",
        'page callback' => 'datadictionary_variables',
        'page arguments' => array("ALL"),
        'access arguments' => array('access biro content'),
        'menu_name' => 'menu-datadictionary',
        'type' => MENU_NORMAL_ITEM
);

return $items;
```

```
}

/**
 * Load configuration from xml-file
 *
 * @param unknown_type $form
 * @param unknown_type $form_state
 */
function load_datadict_config ($form, &$form_state)
 {

        // Checks if a string is null. If it is, return NULL.
         // If its not, return the string with "'s
        function isnull($str)
        {
                if (is_null($str)) {
                        $ret = "NULL";
                } else {
                        $ret = "'" . $str . "'";
                }
                return $ret;
        }

        // Return array-key for given value in associative array
        function array_key($array, $search_value)
        {
                return current(array_keys($array, $search_value));
        }


        $dbg = "DBG_DATADICT";

        watchdog($dbg,"load_datadict_config START");

        // Delete all values in the tables
        db_query("delete from biro_datadict_enum");
        db_query("delete from biro_datadict");
        db_query("delete from biro_crossref_output");
        db_query("delete from biro_crossref_stratum");
        db_query("delete from biro_crossref");

        // Load the xml-file
        $dom = new DOMDocument();
        $dom->load('sites/default/files/biro/datadictionary.xml');
        watchdog($dbg,"xml-dokument lastet");

        // Loop on items
        foreach ($dom->getElementsByTagname('item') as $item)
        {
                // Element?
                if ($item instanceof DOMElement) {
                        $datadict = array();
```

```
            // Loop on the childnodes
            foreach (($item->childNodes) as $itemNode)      {
                if ($itemNode instanceof DOMElement) {

                    // If tag is enumcodes then there should be a
subnode with enumcodes
                    if ($itemNode->tagName == 'enumcodes')  {
                        foreach (($itemNode-
>getElementsByTagName('code')) as $enumcode)  {
                            $datadict['enumcodes'][$enumcode-
>getAttribute('val')] = $enumcode->nodeValue;
                        }
                    } else {
                        $datadict[$itemNode->tagName] =
$itemNode->nodeValue;
                    }
                }
            }

            // Create insert-statement for the datadictionary
            $qry = "insert into biro_datadict
(reference,field_name,parameter,datatype,clinicaldefinition,units) values (";
            $qry .= isnull($datadict['reference']) . ",";
            $qry .= isnull($datadict['fieldname']) . ",";
            $qry .= isnull($datadict['parameter']) . ",";
            $qry .= isnull($datadict['datatype']) . ",";
            $qry .= isnull($datadict['ClinicalDefinition']) . ",";
            $qry .= isnull($datadict['Units']) . ")";

            db_query($qry);

            // Create insert-statement for the enumcodes if the datatype is
enumerated
            if ($datadict['datatype'] == 'Enumerated')
            {
                foreach($datadict['enumcodes'] as $itm)
                {
                    $qry = "insert into biro_datadict_enum ( reference,
enum_code , value) values (";
                    $qry .= isnull($datadict['reference']) . ",";
                    $qry .=
isnull(array_key($datadict['enumcodes'],$itm)) . ",";
                    $qry .= isnull($itm) . ")";
                    db_query($qry);
                }
            }
        }
    }

    // Load the xml-file
    $dom = new DOMDocument();
```

```
        $dom->load('sites/default/files/biro/WP7CrossReference.xml');

        // Loop on items
        foreach ($dom->getElementsByTagname('Indicator') as $indicator)
        {
                // Element?
                if ($indicator instanceof DOMElement) {
                        $crossref = array();

                        // Loop on the childnodes
                        foreach (($indicator->childNodes) as $indicatorNode)        {
                                if ($indicatorNode instanceof DOMElement) {

                                // If tag is enumcodes then there should be a subnode with
enumcodes
                                        if ($indicatorNode->tagName == 'Stratum') {
                                                $crossref['Stratum'][] = $indicatorNode-
>nodeValue;
                                        } else if ($indicatorNode->tagName == 'Output') {
                                                $crossref['Output'][] = $indicatorNode-
>nodeValue;
                                        } else {
                                                $crossref[$indicatorNode->tagName] =
$indicatorNode->nodeValue;
                                        }
                                }
                        }

                        $qry = "insert into biro_crossref
(chapter,target,name,indicatortext,numerator,denominator,source,algorithmcalculation,
algorithmoutput) values (";
                        $qry .= isnull($crossref['Chapter']) . ",";
                        $qry .= isnull($crossref['Target']) . ",";
                        $qry .= isnull($crossref['Name']). ",";
                        $qry .= isnull($crossref['IndicatorText']). ",";
                        $qry .= isnull($crossref['Numerator']). ",";
                        $qry .= isnull($crossref['Denominator']). ",";
                        $qry .= isnull($crossref['Source']). ",";
                        $qry .= isnull($crossref['AlgorithmCalculation']). ",";
                        $qry .= isnull($crossref['AlgorithmOutput']). ")";
                        db_query($qry);

                        if (!is_null($crossref['Stratum'])) {
                                foreach ($crossref['Stratum'] as $stratum) {
                                        $qry = "insert into biro_crossref_stratum
(chapter,stratum) values (";
                                        $qry .= isnull($crossref['Chapter']) . ",";
                                        $qry .= isnull($stratum) . ")";
                                        db_query($qry);
                                }
                        }
```

```
                              if (!is_null($crossref['Output'])) {
                                        foreach ($crossref['Output'] as $output) {
                                                  $qry = "insert into biro_crossref_output
(chapter,output) values (";
                                                  $qry .= isnull($crossref['Chapter']) . ",";
                                                  $qry .= isnull($output) . ")";
                                                  db_query($qry);
                                        }
                              }
                    }
          }

}


/**
 * Implement hook_admin
 *
 * @return unknown
 */
function datadictionary_admin() {
          watchdog("DEBUG","datadictionary_admin");

          $form['Load'] = array(
                    '#type' => 'submit',
                    '#value' => 'Load datadictionary-file',
                    '#submit' => array('load_datadict_config'),
          );

          return system_settings_form($form);
}
```