



**B.I.R.O.**

**Best Information through Regional Outcomes**

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

## **WORK PACKAGE 6: DATABASE ENGINE**

### **DELIVERABLE D6.2**

### **DATABASE ENGINE UPDATE**

**VERS. 0.1**

**WP LEADER: UNIVERSITY OF PERUGIA**



Dept. of Internal Medicine

Via E. dal Pozzo

06126 Perugia Italy

## **Authors**

Valentina Baglioni,  
University of Perugia  
e-mail: [valentina.baglioni@alice.it](mailto:valentina.baglioni@alice.it)

Pietro Palladino,  
University of Perugia  
e-mail: [pietropalladino@gmail.com](mailto:pietropalladino@gmail.com)

## **Address for correspondence**

BIRO Coordination Centre  
Via E. dal Pozzo 06126 Perugia – ITALY  
Ph/Fax. +39 075 5727627  
Email: [biroeu@unipg.it](mailto:biroeu@unipg.it)

## **Project Website**

<http://www.biro-project.eu>

## Index

<b>1. SUMMARY OF WORK PACKAGE 6 UPDATES .....</b>	<b>4</b>
<b>2. BIRO ADAPTOR UPDATE.....</b>	<b>5</b>
2.1. ADAPTOR GENERAL DESIGN.....	5
2.2. DATA SOURCE CONNECTION IMPROVEMENTS .....	5
2.2.1 Importing data from CSV files.....	5
2.2.2 Importing activity data.....	6
2.3. CONFIGURATION IMPROVEMENTS.....	6
2.3.1 Mapping local data to BIRO format .....	6
2.4. NEW ADAPTOR ARCHITECTURE.....	7
2.5. GRAPHICAL USER INTERFACE.....	12
<b>3. BIRO DATABASE MANAGER UPDATE .....</b>	<b>13</b>
3.1. ACTIVITY TABLE.....	13
3.2. MERGE CREATOR .....	13
3.3. GRAPHICAL USER INTERFACE.....	13
3.4. ARCHITECTURE FACILITIES .....	14
<b>4. BIROBOX.....</b>	<b>17</b>
4.1. BIROBOX SETUP .....	18
4.1.1 Setup on Windows platforms.....	18
4.1.2 Setup on Linux Platform .....	20
4.2. USING BIROBOX .....	22
4.2.1 Configuring and running BIRO Adaptor.....	23
4.2.2 Configuring BIRO Database and Database Manager .....	28
4.2.3 Configuring and running BIRO Statistical Engine.....	29
4.2.4 Configuring and running the Communication Software .....	30
4.3. BIROBOX TEST.....	31
<b>5. CONCLUSION .....</b>	<b>34</b>
<b>6. REFERENCES.....</b>	<b>35</b>
<b>APPENDIX A – BIRO ADAPTOR 2 SOURCE CODE.....</b>	<b>36</b>
<b>APPENDIX B – BIRO DATABASE MANAGER SOURCE CODE.....</b>	<b>36</b>
<b>APPENDIX C – BIROBOX SOURCE CODE.....</b>	<b>36</b>

---

## 1. Summary of Work Package 6 updates

---

The BIRO Work Package 6 is aimed at developing the *BIRO Database Engine*.

The *BIRO Database Engine* is a software tool that allows the user to connect to the local data source, to extract relevant data for BIRO purposes and finally to store them into a local BIRO database specifically defined to simplify the work of the Statistical Engine.

The BIRO Database Engine was deeply modified since its first delivery [6] in June 2007.

This deliverable represents an update to the previous version of Database Engine aimed at improving its flexibility in the connection with data sources and its usability. Since the Database Engine is in charge of mapping local fields to the BIRO format and exporting local data into XML documents, the work package 6 ties in closely with the BIRO Data Dictionary and the BIRO Common Dataset. Therefore all updates done within these two work packages have been mirrored by the Database Engine.

The update process consisted of the following:

- development of an improved mapping routine which allows Adaptor to import data fields even if they are not fully compliant with the BIRO standards
- development of a new import routine which allows the Adaptor to extract data not only from databases but also from text files
- development of a new import routine which allows Adaptor to read not only patients' profiles and episodes data from the merge table but also the patients' movements with respect the data centre from a special activity table
- improvement of the Adaptor exporting routine to comply with the revised Data Dictionary
- improvement of the Database Manager importing routine to comply with the revised Data Dictionary
- design and development of BIROBox, a graphical user interface to facilitate the configuration and use of Adaptor, Database Manager and also the other software tools representing the local BIRO environment, i.e. the Statistical Engine and the Communication Software

## **2. BIRO Adaptor update**

---

BIRO Adaptor has been designed to allow the user to connect to a local data source, to retrieve significant data for BIRO purposes and finally to write them down in the form of XML files using the BIRO XML Schema.

### ***2.1. Adaptor general design***

The general design of Adaptor, which has not been changed, consists of the following parts:

1. Data source connection: read data from a source
2. Configuration: configure what to export and how to export
3. Data exporter: stream data from database to the xml file

Most significant updates have been done within the sections of data source connection and configuration.

### ***2.2. Data source connection improvements***

In the data source connection two major updates have been done: the possibility to read data from CSV files and to import data related to the “activity” of patients, namely their location with respect to the collecting centres within different time intervals.

#### *2.2.1 Importing data from CSV files*

Thanks to the JDBC layer, Adaptor is able to connect and retrieve data from many different database engines provided we have the JDBC Java driver (available for many DBMS such as PostgreSQL, MySQL, SQL Server, Oracle, and so on...). The new version of Adaptor is also able to connect and retrieve data from a simple CSV file. This feature is very useful because it allows the user to retrieve data from a wider range of data sources, including spreadsheets and all statistical software tools (e.g. SPSS) that are able to export data from the proprietary format into the CSV format.

When the user provides Adaptor with data in CSV format, Adaptor parses the document, detects each value and finally sets up a new table within an internal HSQL database [1]. This table, which is similar to the CSV file, is indispensable in order to execute the queries that retrieve and process data to be written in the form of XML files.

### *2.2.2 Importing activity data*

In the previous version of Adaptor, only clinical data (episodes and measurements) and patients' profiles (date of birth, date of diagnosis, gender,...) could be imported through the so called "merge table". This table contains a column for each data type and a row for each patient episode.

In the current version of Adaptor it is possible, but not mandatory, to import a wider range of information related to the location and movement of each patient with respect to data collecting centres: the date when the patients were inserted for the first time in the registry of the centre and the reason (birth, diagnosis, transfer from a different centre), the date when the patients exited from the centre and the reason (death, transfer towards a different centre, lost to follow-up).

This kind of information is extremely useful from a statistical point of view because it allows defining precisely the set of patients in the centre within a specific time interval and therefore it allows a better calculation of BIRO indicators.

Data related the activity of patients, if available, should be provided to the Adaptor in the form of the following table, called Activity Table:

{patient ID, start date},start\_reason,end\_date,end\_reason}

The couple Patient ID and start date represents the primary key of the table. Two different records with the same starting date related to the same patient are not allowed. The same patient may appear in more than one record because it is possible for patients to have one continuous or several disjointed periods of activity based on their diagnosis dates, location of residence or follow up status

Also the activity table can be imported from a database or from a CSV file, as it happens for the merge table.

## **2.3. Configuration improvements**

In the Configuration section a newer and more complex data mapping procedure has been inserted.

### *2.3.1 Mapping local data to BIRO format*

In the previous version of Adaptor the user could specify values for static fields in the BIRO Data Set (Site Header and Site Profile) in a plain textual configuration file. The user could also specify a simple mapping procedure by writing, in the same

configuration file, equivalences between the columns' names in the merge table and BIRO data set names as reported in the example below.

```
# Profile Fields
[Profile]

PAT_ID=id_CND
TYPE_DM=TIPODIAB
SEX=SESSO
DOB=NASCITA
DT_DIAG=DATADIAG

# Episode Fields
[Episode]

EPI_DATE=EPIDATE
WEIGHT=WEIGHT
HEIGHT=HEIGHT
```

**Box 1: Example of BIRO Adaptor Configuration File**

In the second version of Adaptor, the mapping procedure was refined.

The goal is always the same: to allow a wider range of data source to be imported and used for BIRO purposes.

Now, for each numeric field the user can specify the unit of measurement adopted in the local data set, by choosing it within a predefined set of available units. Once the local units are defined, Adaptor is able to execute all the needed equivalences to transform local values in the BIRO format.

For each date field is now possible to specify singularly the local pattern. The user can choose the format in use among a set of predefined formats or if any of them is suitable, he can also define a customized pattern. This practically means that every date format can be imported by Adaptor.

Finally for enumerated fields the user can map every single local value to the expected BIRO value. Currently the local value can be a customizable text string, a null value or a regular expression. As future improvement it will be possible to define a mapping between enumerated values and numeric ranges in simple way. At the moment the only way to define numeric ranges is to define a complex regular expression but practical tests have demonstrated that this can be too difficult even for experienced users.

## ***2.4. New Adaptor architecture***

This section describes the consequences of the updates described in the previous paragraphs to the architecture and implementation of BIRO Adaptor.

BIROAdaptor2 is the core class of BIRO Adaptor architecture. First BIROAdaptor2 establishes a connection with the local database using the appropriate driver set in the BIROAdaptorConfigurationFile. Then BIROAdaptor2 initializes the output file through the FileWriter class which opens the necessary stream towards a new ZIP file. This file will contain the XML files of the patients. Finally, BIROAdaptor2 manages the data writing into each XML file: it asks the DBStreamer to execute the query retrieving data from the merge table and the activity table; it scans the ResultSet record by record and creates the corresponding node into the current XML file; it closes the current file and switches to a new one as soon as a new patient is found in the ResultSet. FileWriter is responsible for writing nodes of XML files related to profile fields, activity fields and episode data fields.

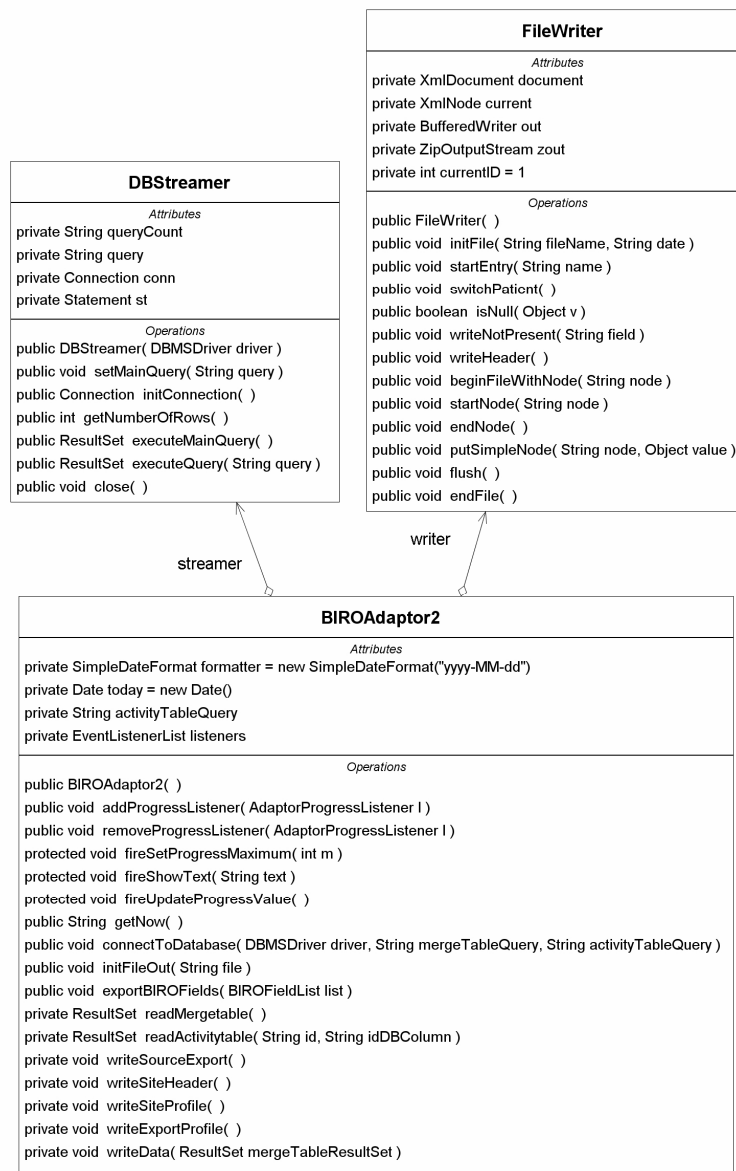
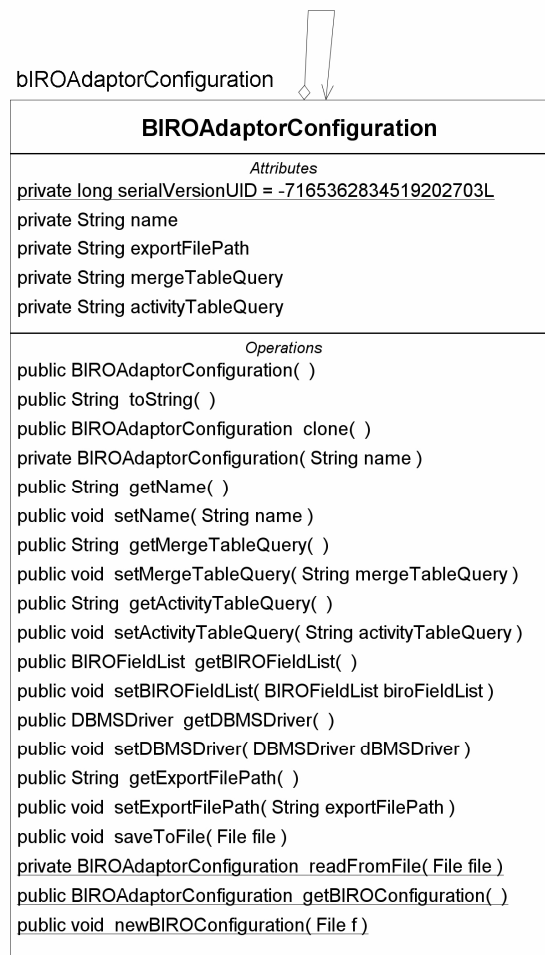


Figure 1- Class diagram of BIROAdaptor architecture

BIROAdaptorConfiguration class contains the following information for the proper functioning of BIROAdaptor:

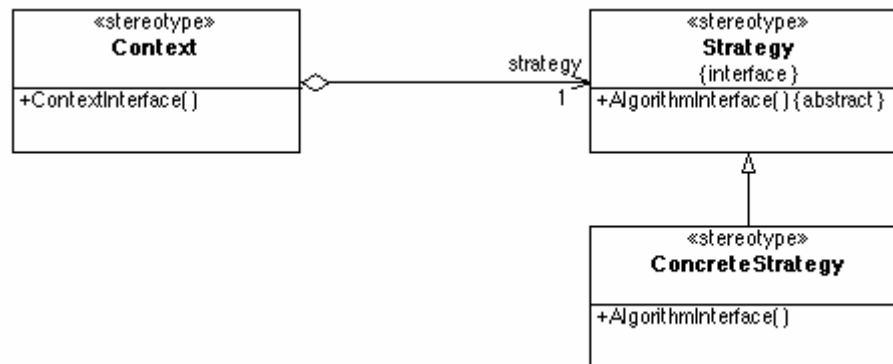
- **dbMSDriver**: all the details to be used to connect to the local data source
- **exportFilePath**: a String representing the complete path of the XML export ZIP file
- **mergeTableQuery** and **activityTableQuery**: two String representing the SQL query to be executed to retrieve the merge table and the activity table from the local source. They can be simple select queries if those tables are already present in the local database or more complex queries if they need to be retrieved by joining other tables.
- **biroFieldList**: the list of BIRO fields with all the necessary information on mapping saved by the user



**Figure 2: UML representation of BIROAdaptorConfiguration class**

The new mapping features of the second version of Adaptor could be obtained through the application of Strategy Pattern [2]. This is a particular software design

pattern, whereby algorithms can be selected at runtime. The strategy pattern is useful for situations where it is necessary to dynamically swap the algorithms used in an application. The strategy pattern is intended to provide a means to define a family of algorithms, encapsulate each one as an object, and make them interchangeable. The strategy pattern lets the algorithms vary independently from clients that use them.



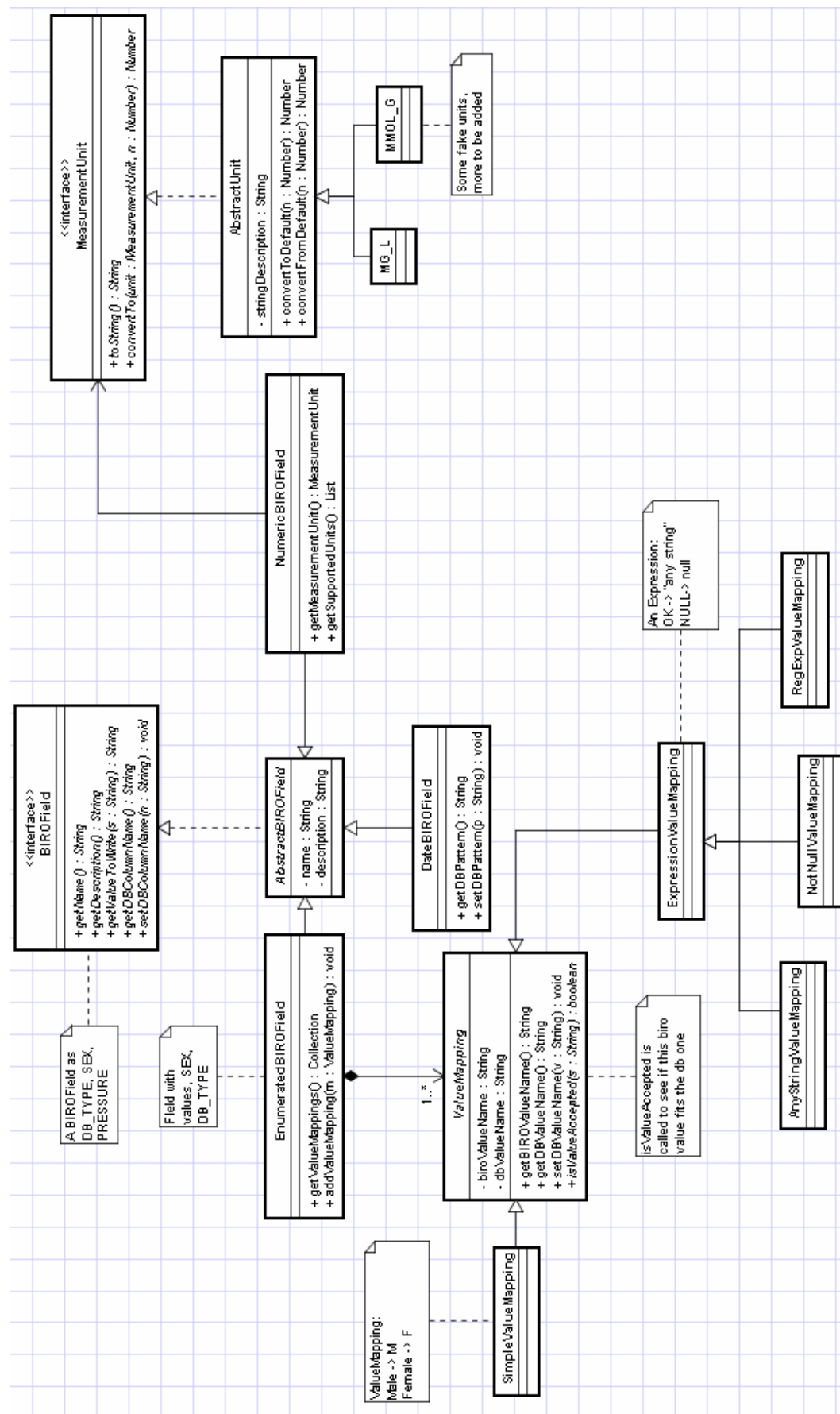
**Figure 3 UML diagram for Strategy Pattern**

The Strategy Pattern was extensively applied to Adaptor architecture as shown by the UML diagram represented in Figure 4:

The AbstractBIROField class represents characteristics and methods common to all BIRO fields. Every different kind of BIRO Field is represented by a different class (EnumeratedBIROField, DateBIROField, SimpleBIROField, NumericBIROField) which extends the AbstractBIROField class. For each different BIRO field, a new class has been created which extends the appropriate field type and contains the name, description and specific characteristics depending on the field type (supported unit, default values, etc.). Every BIRO field implements its own strategy to parse the local values, do the required mappings and return the right value to be written into the nodes of XML files. Each unit of measurement is represented in a different class which extends the AbstractUnit class and is able to do math conversion to and from the default unit. Each mapping strategy for enumerated value is represented in a different class extending the ValeMapping class.

The application of Strategy pattern allows the architecture to be highly extendible: more fields, units of measurements and strategies for enumerated fields can be easily added.

A possible future improvement could be the possibility to add new unit of measurements at runtime, which is not possible now.



## ***2.5. Graphical User Interface***

To configure and use the Adaptor a set of wizards has been implemented and included in the BIROBox, which is a comprehensive graphical user interface for the whole BIRO local system. In the first release Adaptor could be executed from the console and its behaviour was controlled through configuration file, read at start-up. In the current version, the increase of mapping choices and configuration features has made a graphical user interface necessary. It was not feasible anymore to configure the tool through a plain text file which would require a big effort from the user and a deep knowledge of the configuration format.

Now the data source configuration can be made by filling a form instead of writing text; the mapping of BIRO fields with local fields can be made by simply choosing the appropriate options from a predefined list; the export can be started by a click on a button instead of writing a command into the console

### 3. BIRO Database Manager Update

---

Few changes have been done to BIRO Database Manager since the release of first version.

Database Manager has been updated in order to comply with the changes occurred to the Common Dataset, Data Dictionary and XML Metadata Dictionary and therefore to comply with the changes occurred to the format of Adaptor output.

#### 3.1. Activity table

Since a new section called Activity Data has been added to the BIRO Export XML schema to include information regarding the movements of the patients with respect to the collecting centre, the Database Manager has been updated accordingly. A new table, called Activity Table has been added to the local BIRO Database structure, where Database Manager can store the new set of information.

#### 3.2. Merge creator

Merge Creator is a new feature which has been added to the Database Manager. Merge Creator is a small stand alone internal application which resides within the package of Database Manager. It allows to process the BIRO local database and reconstruct a new table similar to the merge table, starting from the standard BIRO structure.

Merge Creator is able to execute a set of queries to the BIRO local Database in order to obtain the two following tables:

- Profile\_wide: this table contains a column for each profile field in the BIRO Dataset; the primary key is represented by the Patient ID.

Profile\_wide {patient\_id, bob, dt\_diag, pat\_id, sex, type\_dm}

- Episode\_wide: this table contains a column for each Episode field in the BIRO Dataset; the primary key is represented by the Patient ID and the episode date.

Episode\_wide {patient\_id,episode\_date, [episode field]}

#### 3.3. Graphical User Interface

To configure and use the Database Manager a set of wizards has been implemented and included in the BIROBox, which is a comprehensive graphical user interface for the whole BIRO local system. In the first release, Database Manager could be

executed from the console and its behaviour was controlled through a configuration file, read at start-up.

In the current version all the information needed to run the Database Manager can be specified by filling some appropriate forms and in particular:

- the path of the ZIP file containing the output of Adaptor
- the JDBC configuration for the BIRO local Database

The import of local data into the BIRO database can be started by a click on a button instead of writing a command into the console.

### **3.4. Architecture facilities**

Updating BIRO Database Manager is a quite easy process because of the design of its architecture. BIRO Database Manager contains a stand alone application, named BIROClassGenerator, that reads the XML schema and automatically generates the source code of Java classes related to the elements in the XML schema. It is very useful for the development and maintenance of BIRO Database Manager because if changes to the XML schema are needed, the structure of Java classes can be immediately updated accordingly. The BIROClassGenerator uses the Castor [3] feature named SourceGenerator. It requires as input the BIRO XML schema and a binding file, which is an XML file used to specify some customized constraints regarding class names and data types to be respected while generating Java source code.

Here follows an example of how the SourceGenerator works. In Box 2 and Figure 5: UML diagram for SiteHeader class there are, respectively, an abstract of the XML schema regarding Site Header information and the UML representation of the generated SiteHeader Java class. SourceGenerator transforms all the elements in attributes of SiteHeader class, then it creates the related getter and setter methods and finally it adds methods for marshalling, unmarshalling and validation.

The mapping between Java Object Model and Relational Model is specified by means of one or more XML documents named mapping files. These files are used to create the database architecture and to store at the right place the unmarshalled Java Objects. Since there is not any software facility that creates mapping files automatically, they should be updated manually whenever the XML schema changes.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="0.4">
<xsd:element name="ECDataSourceExport">
```

```

<xsd:complexType>
  <xsd:sequence>
    <xsd:element name="SiteHeader">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="DateHeaderInformationChecked"
type="xsd:date"/>

          </xsd:element>
          <xsd:element name="DS_ID" type="DataSource" id="BIR0002">
          </xsd:element>
          <xsd:element name="DS_WEBSITE" type="xsd:string" id="BIR0106"
minOccurs="0">
          </xsd:element>
          <xsd:element name="DS_ADDRESS_1" type="xsd:string" id="BIR0107">
          </xsd:element>
          <xsd:element name="DS_ADDRESS_2" type="xsd:string" id="BIR0108">
          </xsd:element>
          <xsd:element name="DS_ADDRESS_3" type="xsd:string" id="BIR0109"
minOccurs="0">
          </xsd:element>
          <xsd:element name="DS_ADDRESS_4" type="xsd:string" id="BIR0110"
minOccurs="0">
          </xsd:element>
          <xsd:element name="DS_POST_CODE" type="xsd:string" id="BIR0111"
minOccurs="0">
          </xsd:element>
          <xsd:element name="DS_COUNTRY" type="xsd:string" id="BIR0101">
          </xsd:element>
          <xsd:element name="DS_C_CONTACT" type="xsd:string" id="BIR0112">
          </xsd:element>
          <xsd:element name="DS_C_EMAIL" type="xsd:string" id="BIR0113">
          </xsd:element>
          <xsd:element name="DS_T_CONTACT" type="xsd:string" id="BIR0114">
          </xsd:element>
          <xsd:element name="DS_T_EMAIL" type="xsd:string" id="BIR0115">
          </xsd:element>
          <xsd:element name="HeaderComments" type="xsd:string"
minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:ComplexType>
</xsd:element>
</xsd:schema>

```

**Box 2: XML Schema - Site Header**

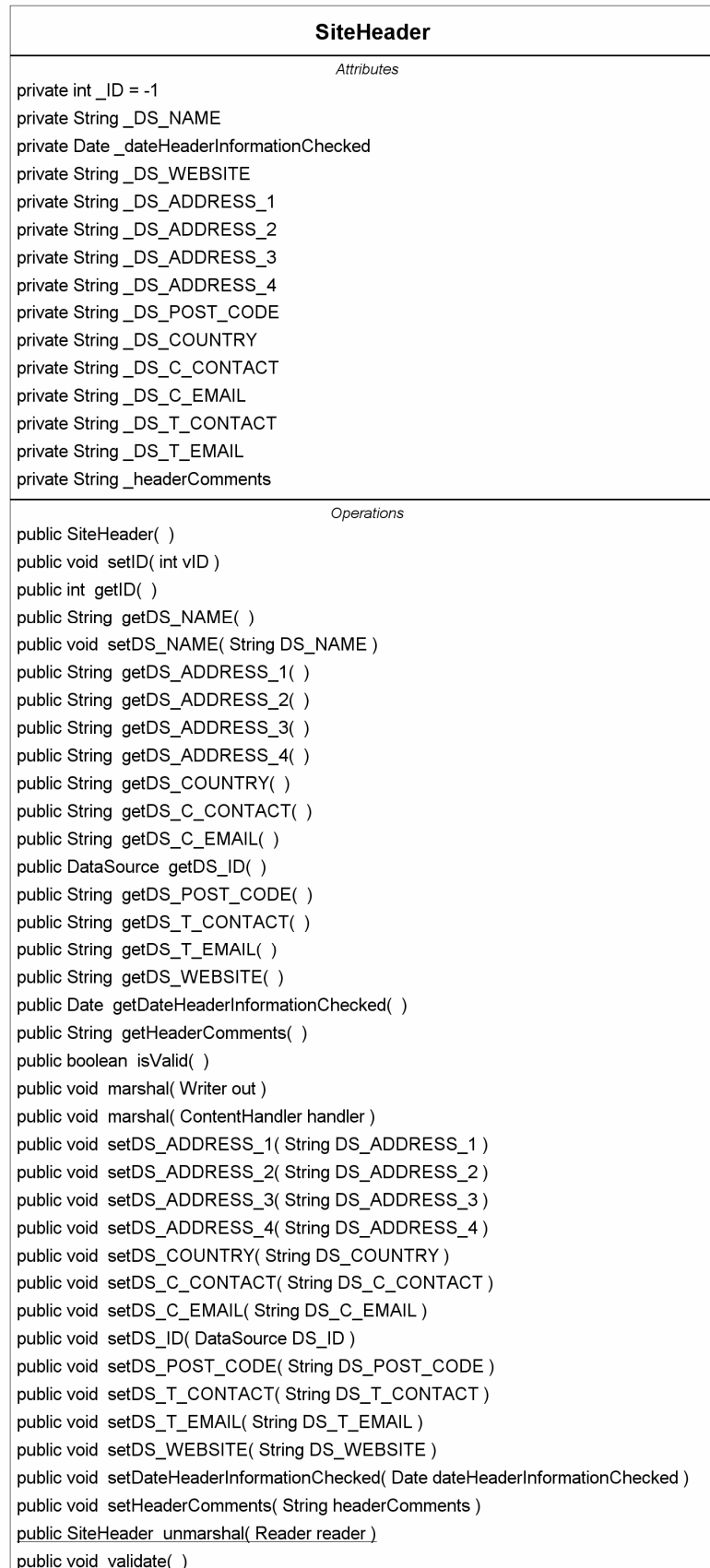


Figure 5: UML diagram for SiteHeader class

## 4. BIROBox

The BIROBox is a Java application aimed at representing the Graphical Interface for the whole local BIRO system and its software tools.

In their previous release both BIROAdaptor and BIRODatabaseManager could be configured through a plain text file and could be only executed separately from the console. In the current version, the increase of complexity in the configuration and use of Adaptor and Database Manager has made a graphical user interface necessary.

In order to improve the usability of Database Engine tools, the BIROBox GUI was developed. Then BIROBox was conveniently adopted as user interface for all software tools representing the local BIRO environment, including BIROStatisticalEngine and BIROCommunicationSoftware. Now BIROBox represents a comprehensive approach for the implementation and use of BIRO system.

A screenshot of BIROBox can be seen in Figure 6.

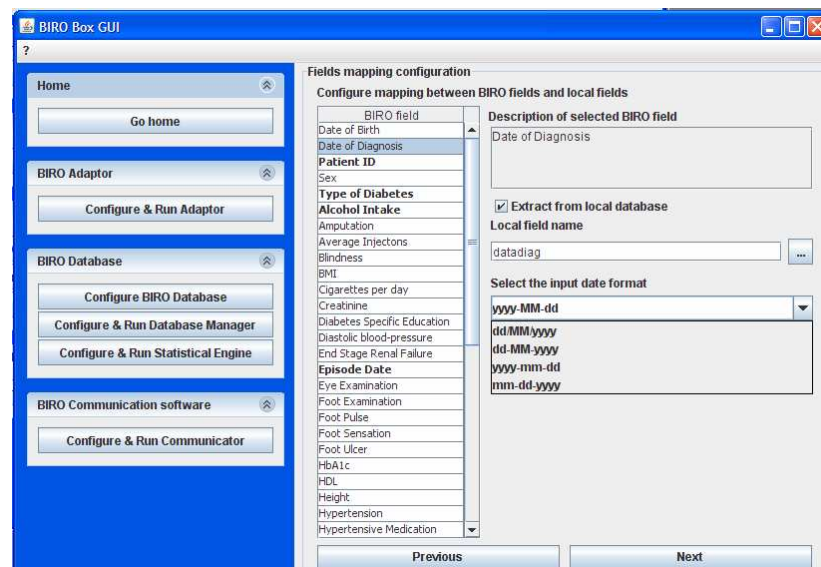


Figure 6: Screenshot of BIROBox

As the main point of entry to the usage of BIRO is represented by the BIROBox, here we will focus on its setup, launch, and usage.

## 4.1. BIROBox setup

### 4.1.1 Setup on Windows platforms

A one-click setup file has been developed by JOANNEUM Partner to simplify the install procedure of BIROBox on Windows environment. The setup file contains all prerequisite software and will automatically launch the setup procedure if the user wants to install them:

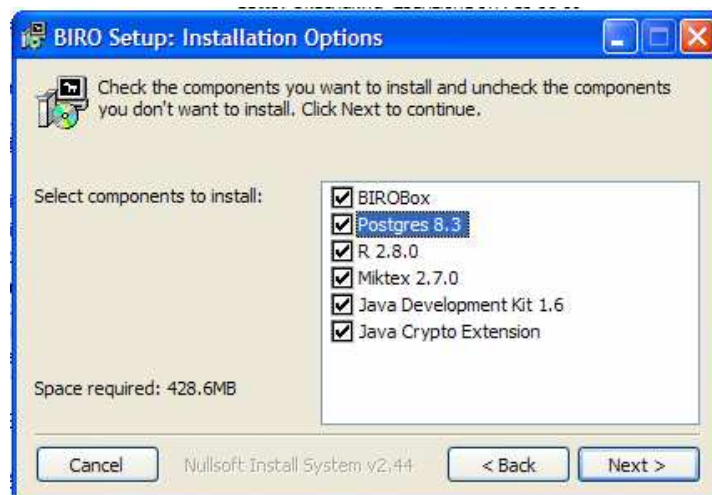
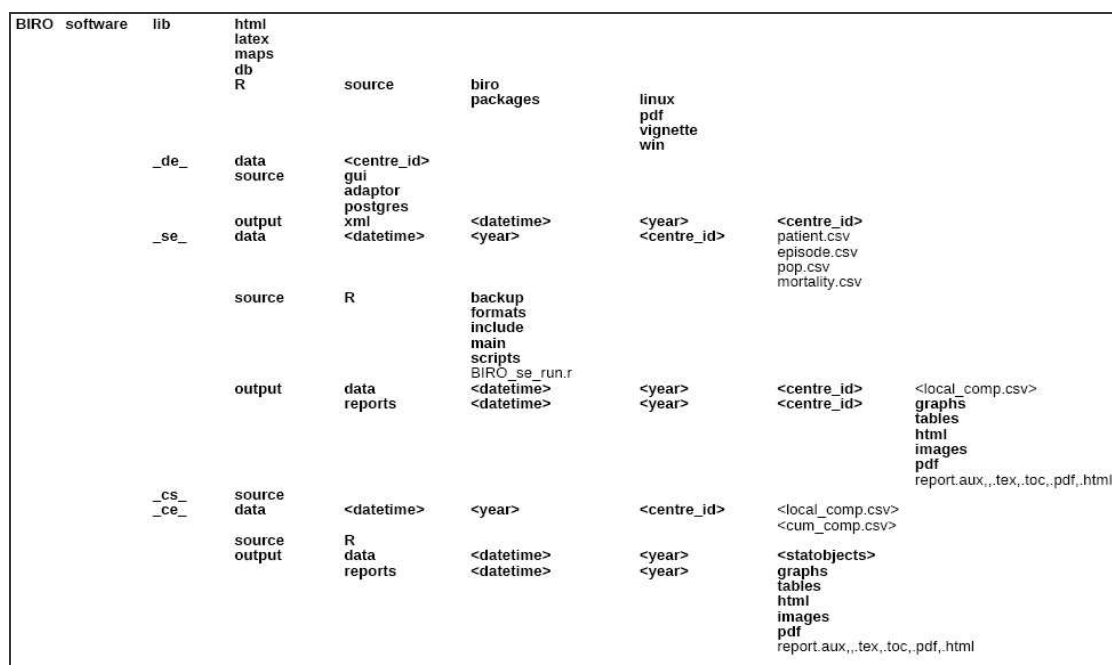


Figure 7: BIROBox setup routine for Windows

Prerequisite software is the following:

- PostgreSQL 8.3.7: this is an open source DBMS, the user may want to install it if any other DBMS is available to host the biro database.
- R: is the statistical framework on top of which the BIRO statistical engine was built. The statistical engine is optimized to work with the R version 2.8.0. Also later versions of R could be used but in that case the user should manually install all the required R packages (see section 4.1.1.1)
- Miktex 2.7.0: this is a software utility that allows BIROBox to print formatted reports in PDF files
- Java Development Kit 1.6: this is the Sun Microsystem product containing the Java Virtual Machine. Later versions of JDK are also suitable for BIROBox
- Java Crypto Extension (JCE) is an extension of the JDK that provides a framework and implementations for encryption and message authentication [4]

The BIROBox setup routine will create the BIRO System directory structure shown in Figure 8, which will contain all the source code and libraries and documentation related to the BIRO System.



**Figure 8: BIRO System Directory Structure**

Some special folders needs to be highlighted as they will contain the outputs of the BIROBox:

- *BIRO/software/\_de\_/data*: default folder hosting the BIRO Export XML files produced by Adaptor
- *BIRO/software/\_se\_/output/data*: folder hosting local statistical objects produced by the Statistical Engine
- *BIRO/software/\_se\_/output/report*: folder hosting local statistical report produced by the Statistical Engine

The setup routine will also set up automatically all the required environment variables. Once the setup is completed, a shortcut will be created in the Programs' menu to easily run and uninstall BIROBox.

BIROBox can be also started by clicking on the "runBIROBox.bat" file located at the root of the newly created BIRO environment.

#### 4.1.1.1 *Using BIROBox with different R versions*

If you want to use BIROBox with different R versions, you should download and install the following packages manually:

- mingw
- Cairo
- rJava
- DBI
- RJDBC
- lattice
- rmeta
- sp
- maptools
- Hmisc
- Epi
- epicalc

R packages can be easily installed through the command `install.packages('<packageName>')` or through the appropriate package manager wizard.

#### 4.1.2 *Setup on Linux Platform*

The setup routines for BIROBox is not available for Linux platform as for Windows platform. The BIROBox version for Linux is released as a simple tar.gz archive. Here follows the detailed description of the necessary settings.

The following prerequisite software should be installed:

- A DBMS is required to host the BIRO database. If any database is present, we suggest to download and install *PostgreSQL 8.3.7* ([www.postgresql.org](http://www.postgresql.org))
- *R* statistical software (<http://www.r-project.org/>). Version 2.8.0 is recommended but also later versions of R could be used.
- *Java Development Kit* 1.6 or later. Both Open JDK (<http://openjdk.java.net/>) and Sun Microsystems' JDK (<http://java.sun.com>) are suitable.

Once you have downloaded the BIROBox tar.gz archive from the BIRO web site, please extract it in the folder where you want it to be installed.

Then be sure that the following environment variables are set:

- Set the *BIRO\_HOME* environment variable as *BIRO\_HOME=<BIRO setup folder>/Biro*
- Set the *R\_HOME* environment variable
- Set the *JAVA\_HOME* environment variable
- Set the *AXIS2\_HOME* environment variable as  
*AXIS2\_HOME=\$BIRO\_HOME\software/\_cs\_/axis2-1.4*
- Update your PATH environment variable by adding the *R\_HOME* at the end of it

Finally the following additional settings should be done:

- Copy the file *\$BIRO\_HOME/software/\_cs\_/resources/bcprov-jdk16-141.jar* to your *\$JAVA\_HOME/jre/lib/ext* folder
- Copy the 2 jar files (*local\_policy.jar* and *US\_export\_policy.jar*) from *\$BIRO\_HOME/software/\_cs\_/resources/jce\_policy-6.zip* to *\$JAVA\_HOME/jre/lib/security* and overwrite existing policies.
- Add the following entry to your *\$JAVA\_HOME/jre/lib/security/java.security* file:  
*security.provider.N+1=org.bouncycastle.jce.provider.BouncyCastleProvider* where N is the highest number available in the list of SecurityProviders
- Download and install the following R packages:
  - *Cairo*
  - *rJava*
  - *DBI*
  - *RJDBC*
  - *lattice*
  - *rmeta*
  - *sp*
  - *maptools*
  - *Hmisc*
  - *Epi*
  - *epicalc*

R packages can be installed through the execution of the command *install.packages('<packageName>')* from the R shell.

- Copy the library file *libjri.so* from *\$R\_HOME/rJava/JRI* folder into *\$BIRO\_HOME/software/lib/jri*

Once all settings have been done, BIRO Box can be launched through the script *runBIROBoxGUI* located in *\$BIRO\_HOME*.

## 4.2. Using BIROBox

When the user run the BIROBox, the main windows appears as shown in Figure 9.



Figure 9: BIROBox Home Page

All main functions of BIRO system can be accessed by selecting the appropriate button of the button panel located on the left part of the window. The bigger area on the right part of the window is designed to contain the configuration panels for each function.

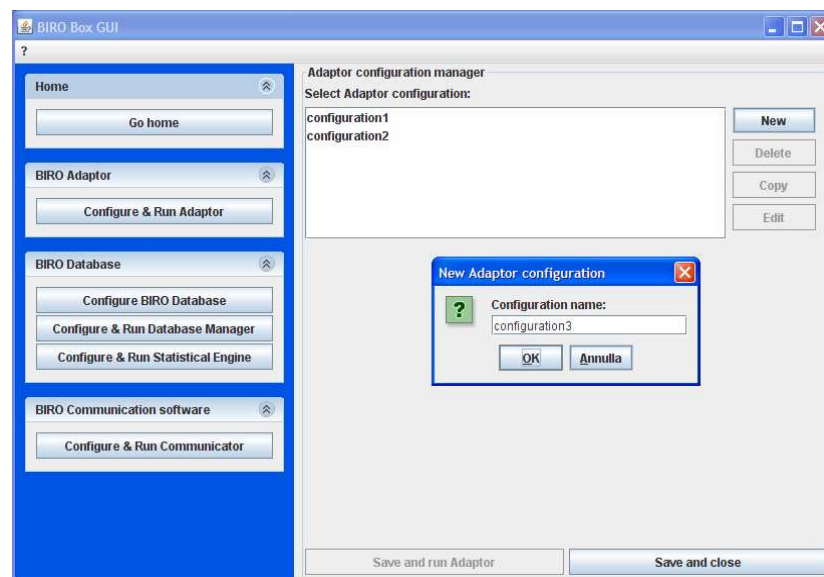
Each BIRO function (data extraction from local source, creation and population of local BIRO Database, print of the statistical report, data sending to the central BIRO system) can be executed separate from others, and steps are triggered by the local user. Alternatively, it is possible to run all tools at once automatically, although the step-by-step approach is preferable since it provides more liberty in the management of the local system. This way, one would import database XML files into the local BIRO database even if those files are produced by a different centre; or if a database is already in the BIRO format, one would choose to skip the execution of the Adaptor and Database Manager; in case the user is interested only in printing the local report (but not sending data to the central system) he/she would launch the Statistical Engine at any time without running the Communication Software.

#### 4.2.1 Configuring and running BIRO Adaptor

Clicking on the “Configure and Run Adaptor” button allows the user to manage the Adaptor configurations and run the Adaptor on top of the selected configuration.

Multiple configurations are allowed for the Adaptor because the user may have multiple data sources.

The user can create as many configurations as he wish. Each configuration can be deleted, copied or edited.



**Figure 10: Adaptor - Configuration Manager**

##### 4.2.1.1 Setting the connection to the local data source

The first step for configuring the Adaptor is setting the connection to the local data source, a database or a CSV file. If a CSV file, the user must specify the filename and the separator format.

In case of a database, the user must specify the connection and login details: driver type, host and port, database name, user name and password. If none of the drivers listed is suitable for the local DBMS, the user can then add a custom DBMS driver by clicking on the “+” button .

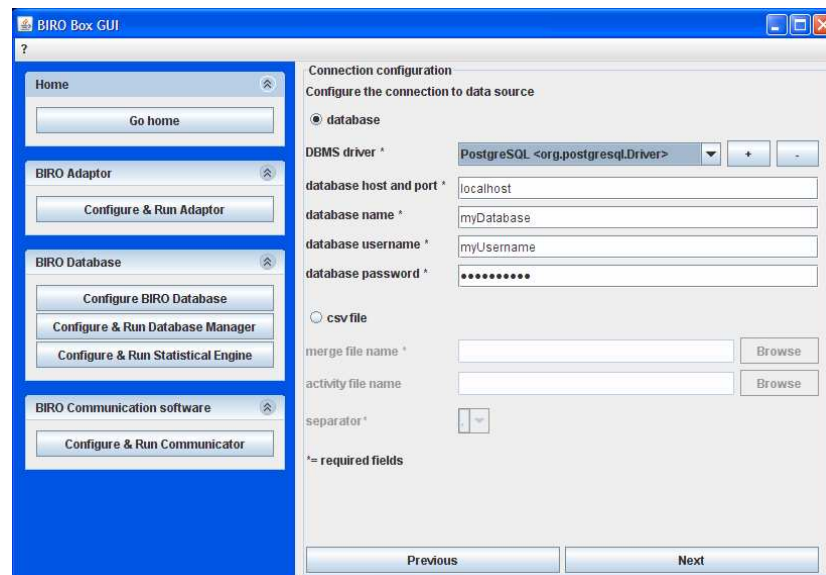
In order to create a new DBMS driver the user must specify:

- the DBMS name (e.g. “PostgreSQL”)
- the JDBC driver class name (e.g. “org.postgresql.Driver”)
- the URL pattern with flags for host and port, database name, database username (optional), database password (optional) instead of real values (e.g. “jdbc:postgresql://<hostAndPort>/<databaseName>”)

- the absolute path of the jar file containing the JDBC driver for the local DBMS (e.g. “C:\myFolder\postgresql-8.2-504.jdbc3.jar”)

The creation of a new driver requires some IT proficiency, so it should be done with the assistance of an IT expert.

When clicking on the “next” button, the connection is tested: if something goes wrong and is not possible to establish a connection to the data source, then the user must stop configuring the Adaptor and attempt to repeat the operation with different options.



**Figure 11: Adaptor - Data Source Connection Configuration**

#### 4.2.1.2 *Configuring merge table and activity table*

In the second step, the user must configure the merge table (Figure 12). If the data source is a CSV file or the merge table is already present in the local database, the user has to select the first option and write the merge table name into the text field. By clicking the button on the right, the user may inspect the local database and obtain the complete list of the tables. He/she may choose to import the whole merge table or just a subset of records, through the definition of the start date and end date for episodes.

If the merge table is not present in the local database, the user can create it by writing the appropriate SQL query in the text area.

In the third step, the user has to configure the activity table by specifying its name or the query to obtain it.

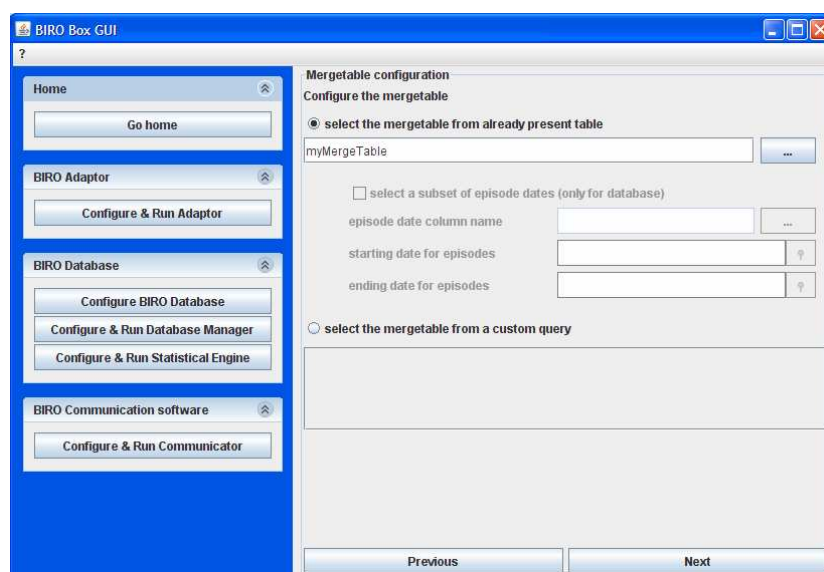


Figure 12: Adaptor - Merge Table Configuration

#### 4.2.1.3 Inserting Data Source profile

In the data source configuration panel (Figure 13) the user has to input static information about the centre (data source ID, clinical and technical contacts...) and about the catchment area (total population, number of diabetologists, nurses, doctors...). Site profile fields will be used by statistical engine to calculate some of the indicators in the BIRO reports

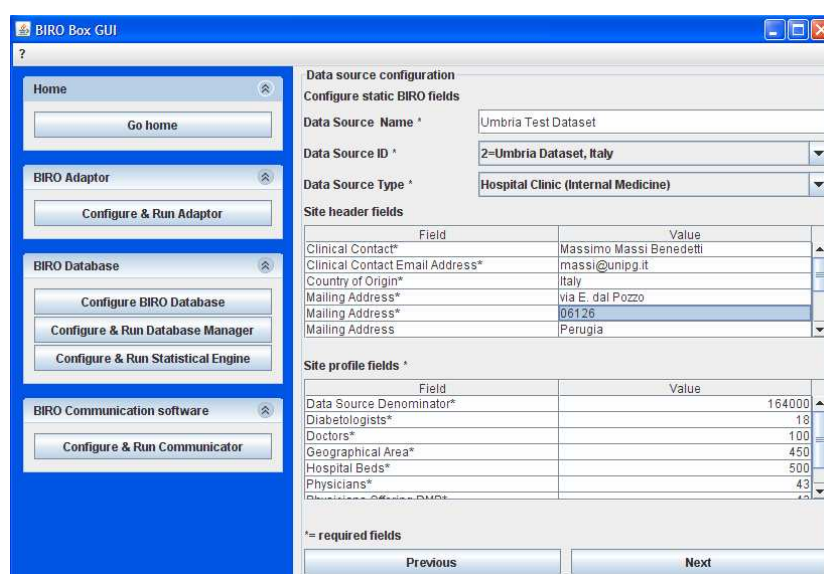


Figure 13: Adaptor - Data Source Configuration

#### 4.2.1.4 Mapping local fields to BIRO fields

Mapping local fields to BIRO fields is probably the most complex aspect of configuring the Adaptor (Figure 14).

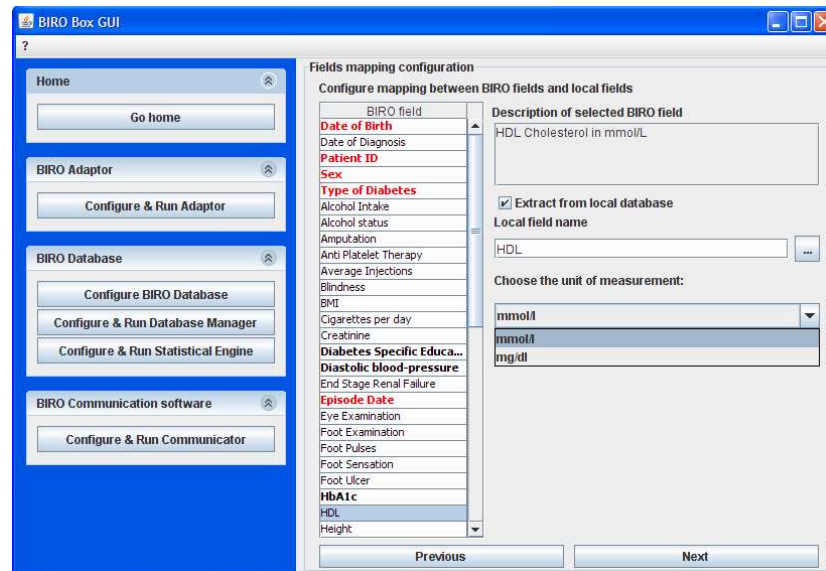


Figure 14: Adaptor: BIRO Field Mapping

All BIRO fields are listed on the left side of the panel. For each of them, the user must specify if the field can be extracted from the local source, the name of the local column and the local data format. Mandatory fields are highlighted with a bold red font. Fields to be extracted are highlighted in bold.

In order to speed up the process of mapping BIRO fields, the user can inspect the local data source by selecting the special button on the right side of the panel.

There are three types of fields, each one requiring a different mapping.

- Date fields: mapping is done by choosing the date format in use in the local data source
- Numeric fields: the user must choose unit of measurement adopted in the local data source. Simple fields, like patient ID and BMI, do not require any mapping.
- Enumerated fields: for each enumerated value the user has to write the correspondent value in local data source. Several choices are possible: null value, any string, null or any string, regular expression, custom text. While the custom text option is used to map straight one-to-one relations, regular expressions can be used to map one-to-many relations.

Example: the mapping of the Type of Diabetes field represents a common example of one-to-one and one-to-many mapping. Let's consider the following situation: the local data records four different values in the type of Diabetes fields:

- 1 = type 1 diabetes;
- 2 = type 2 diabetes;
- G = Gestational diabetes
- M = Monogenic Diabetes

These enumerated values should be mapped to only three BIRO enumerated values

- 1 = type 1 diabetes;
- 2 = type 2 diabetes;
- 3 = other types of diabetes;

In order to map type 1 the user can select the “custom text” option with the string “1” as argument. The same applies for the mapping of type 2 values.

Gestational Diabetes and Monogenic Diabetes should both be mapped to other types of diabetes. This can be done by selecting the “regular expression” option and writing as argument an expression representing the logical OR of values representing gestational and monogenic diabetes i.e. the following expression “G/M”.

The last step of Adaptor Configuration is the output file: the user can select where to save the ZIP file produced by Adaptor. As previously described, the default folder is:

*\$BIRO\_HOME/Biro/software/\_de\_/data*

By going back to the Adaptor configuration manager panel and clicking on the Run Adaptor Button, the process is started and a progress status window is showed. At the end of the process, an XML file per patient is created (see Box 3) and all XML files are compressed in a single big ZIP file.

```
<?xml version="1.0" encoding="cp1252"?>
<!-- BIRO Export File -->
<ECDataExport xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Patient>
    <Profile>
      <ProfileFieldName>DOB</ProfileFieldName>
      <ProfileFieldValue>1935-07-19</ProfileFieldValue>
    </Profile>
    <Profile>
      <ProfileFieldName>PAT_ID</ProfileFieldName>
      <ProfileFieldValue>1165</ProfileFieldValue>
    </Profile>
    <Profile>
      <ProfileFieldName>SEX</ProfileFieldName>
      <ProfileFieldValue>1</ProfileFieldValue>
    </Profile>
    <Profile>
      <ProfileFieldName>TYPE_DM</ProfileFieldName>
      <ProfileFieldValue>2</ProfileFieldValue>
    </Profile>
    <EpisodeData>
      <EpisodeDate>2003-07-01</EpisodeDate>
      <Data>
        <EpisodeFieldName>HBA1C</EpisodeFieldName>
        <EpisodeFieldValue>73.0</EpisodeFieldValue>
      </Data>
    </EpisodeData>
  </Patient>
```

**Box 3: Example of BIRO export XML file**

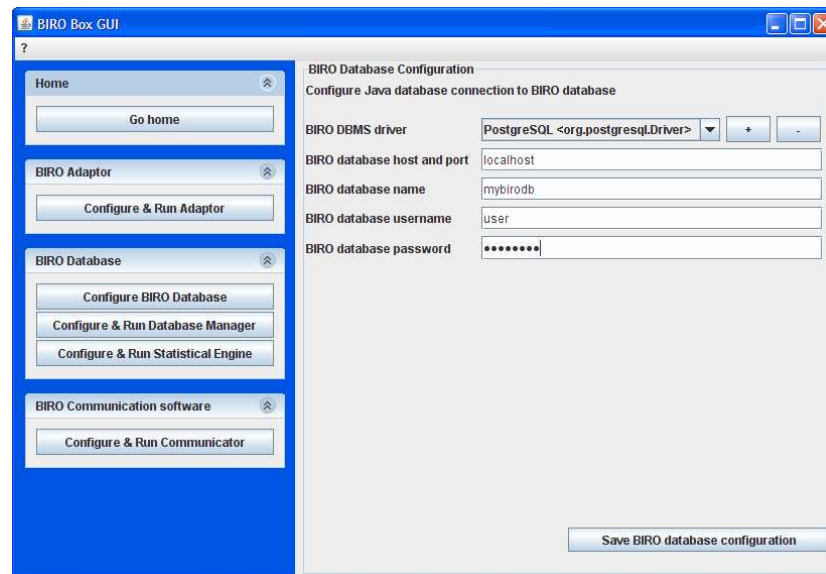
#### 4.2.1.5 *Restrictions on local data format*

Even if Adaptor was not designed with the aim of deciding about the correctness of the local data, it performs some simple tests on the data format to avoid possible errors while executing subsequent steps:

- Adaptor assumes that the `patient_id` and `episode_date` columns represent a primary key for the local data set. Two or more rows with the same values for `patient_id` and `episode_date` are not admitted therefore Adaptor will stop if any duplicate is found. Instead of attempting to correct the error, Adaptor just makes a report containing the whole list of duplicated rows.
- Date of birth, patient id, episode date, sex, type of diabetes are mandatory field because they are stratification factors for most part of indicators calculated in the BIRO report by statistical engine. The Adaptor will not start its process until all of them are set. Since further improvements of statistical engine may allow a dynamic selection of the stratification factors, this requirement could be removed in the future.
- Currently it is not possible to choose the desired Locale settings which would allow the processing of data coming from a country with different rules for date and number formats. Adaptor automatically detects the default locale of the Java Virtual Machine. Localization and internationalization of BIROBox will be a required improvement for the extensibility of the BIRO System to other European or even extra European Countries.

#### 4.2.2 *Configuring BIRO Database and Database Manager*

The configuration of the BIRO local database is necessary to run the BIRO Database Manager or the BIRO Statistical Engine. The BIRO database configuration is similar to the data source connection configuration (see Figure 15): the same url (DBMS Driver, database name, host and port) and login details (username and password) are requested to the user. Once the BIRO Database is set, the user just need to specify the input ZIP folder for Database Manager containing the patients' XML files to be imported.



**Figure 15: BIRO Database Configuration**

#### *4.2.3 Configuring and running BIRO Statistical Engine*

The Statistical engine requires few data to be configured (Figure 16):

- centre ID
- current year
- start year for analysis
- duration (time interval for data analysis)
- reference date
- population file and diabetic population file (CSV)

The latter are fundamental files containing the total population and the diabetic population in the catchment area for the correct calculation of population based indicators. In particular, the total population should be stratified by year, age band, gender (Table 1), while the diabetic population requires strata by year, ageband, type of diabetes and gender (Table 2).

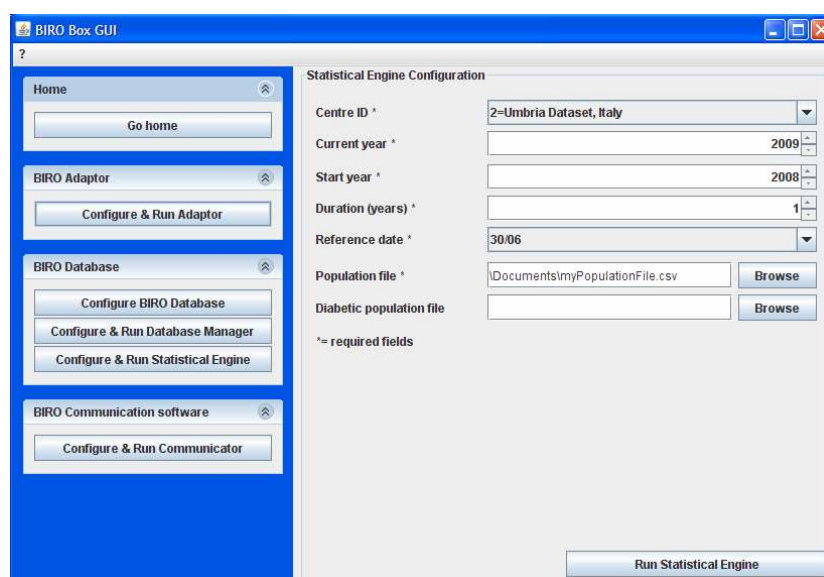


Figure 16: Statistical Engine Configuration

year	ageband	popM	popF	morM	morF
1997	1	19356	18289	8	14
1997	2	18623	17240	3	0
1997	3	18641	17562	5	1
1997	4	19819	18511	4	2

Table 1: BIROBox – Example of population table

year	ageband	typedm	diabM	diabF
1997	1	1	100	90
1997	2	1	201	300
1997	3	1	343	250
1997	4	1	432	300

Table 2: BIROBox- Example of diabetic population table

#### 4.2.4 Configuring and running the Communication Software

Every time the user runs the Statistical Engine, a statistical report is produced in HTML and PDF format, stored in folders named with the current timestamp. The Communication Software panel (Figure 17) shows the complete list of statistical objects created by the Statistical Engine to be transferred to the central engine. For each statistical object, the creation and last sending date are duly specified.

The user should also specify the IP address of the BIRO Central Server, which is not hard coded into the Communication Software.

When the user select one of the statistical objects in the list and clicks the “send” button, the Communication Software creates a compressed folder, sent it to the central server where it will be decompressed and permanently stored in the central database. The Central Server currently does not have any special GUI tool available and is only operated by the BIRO development team.

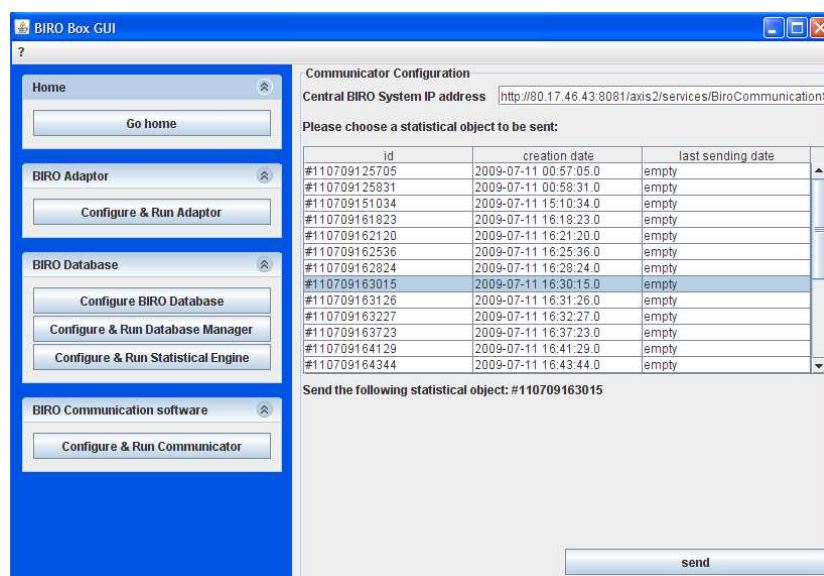


Figure 17: Communication Software Configuration

### 4.3. BIROBox test

In terms of developing a sustainable information system, the BIRO Consortium focused its attention mainly on creating a solution that could be applicable in real life situations, where diabetes data are used and stored on a daily basis. Clearly, much of the success of our initial trial was based on the applicability of the most visible and direct component: the BIROBox. The BIROBox was initially tested on a sample dataset extracted from the Umbria register and on other databases from Cyprus, Malta and Romania during the BIRO Technology Transfer Meeting (Bergen, Norway, 15th-17th January 2009). Then the official test of BIROBox was performed by the whole Consortium on real data sets coming from local sources.

These tests highlighted many positive aspects of the BIRO software and were the occasion to collect suggestions regarding possible improvements on the software from the whole Consortium.

The fact that BIRO was delivered in the form of a single setup file was very appreciated: with a simple double click, the tool allowed to install everything produced by the system, including documentation.

The BIROBox also allowed to fill forms or choose options, as well as inspecting the local data source, or looking up a list of all local tables within the chosen database, and local fields within the selected table. Test users noticed that they inevitably needed to know how to harmonise the local dataset with the BIRO standard, highlighting the fact that using the BIROBox cannot circumvent the need for a detailed knowledge of the system.

A special attention was dedicated to the possibility for the BIROBox to connect with many different data sources. A survey has been conducted within the BIRO Consortium in order to know the technology in use for each local data source. All drivers for the most common DBMS have been added to the driver list of BIRO Adaptor. Although most database drivers were included in the list of those usable for BIRO, the need for a “custom” driver was made clear, and consequently added to the box. However, such customization makes use from a non technical person unfeasible.

The alternative of use of CSV files has been very positively evaluated as a means of bypassing the problem of creating a customized driver, using this format as a *lingua franca* for the transport of files from any format to BIRO.

Finally, there was a good performance of the BIRO Adaptor in terms of execution time: about 5 minutes to transform nearly 100,000 patient's episodes from the local data source to the XML export. Obviously, the process duration showed to depend both on the number of episodes recorded and the number of BIRO fields exported. As suggested by the Consortium, the following aspects of the BIROBox would need to be revised in the future:

- popup help screens are not available for each form, especially those regarding data source properties and warning messages; it is important to prevent the user from inserting wrong entries;
- it is not possible to check the correctness of data mapping in real time: if the user makes a mistake when mapping local field to BIRO field (e.g. a wrong date format or a wrong unit of measurement is chosen), it will be noticed only when running Adaptor. BIROBox should immediately check mapping and send warnings to the user.
- Adaptor stops execution if a record contains wrong or unexpected data; modification is needed so that wrong values are simply discarded and each error is reported into an error log file. This could prevent multiple stops in case of poor quality datasets.

- all BIRO fields are hard coded within the Adaptor source code as java classes: this means that adding new fields to the BIRO Dataset would require modifying and recompiling the whole source code. In a future revision, software must allow more flexibility with respect to the changes on data dictionary.
- although for each numeric BIRO field the user can select the unit of measurement used locally (within a predefined set) and map it automatically into the official BIRO unit, the user cannot add any other unit of measurement. This is because the mapping rules between units are hard coded. The list of units of measurement should be expanded, or let the user specify conversion algorithms.
- the Database Manager process requires long time to be completed because XML files are transformed into Java objects and stored into the local database one-by-one. Speeding up the process would be very important for the user.

## 5. Conclusion

---

A great amount of work has been done from the first version of Database Engine until now. The combined efforts of all Partners have transformed a set of separate software tools in a comprehensive framework, the BIROBox.

We hope that an increasing group of users could be interested in implementing and testing BIROBox since running the BIROBox on top of real data is the only way to test the effectiveness of the BIRO architecture and produce strategic information to support coordinated prevention, integrated care and outcomes management in diabetes across Europe. Acknowledgments should be expressed to all BIRO Partners for their valuable suggestions and feedbacks, which have been essential for the improvement of the software.

## 6. References

---

- [1] HSQLDB (Hyper Structured Query Language Database) is a relational database management system written in Java. More details can be found at <http://hsqldb.org>
- [2] More details about Strategy Design Pattern can be found at [http://en.wikipedia.org/wiki/Strategy\\_pattern](http://en.wikipedia.org/wiki/Strategy_pattern)
- [3] More details about Castor project can be found at <http://www.castor.org/>
- [4] More details on Java Cryptography Extension (JCE) can be found at <http://java.sun.com/javase/technologies/security/>
- [5] BIRO deliverable D6.1 Database Engine can be found at <http://www.biro-project.eu/results.htm>

## **Appendix A – BIRO Adaptor 2 source code**

---

## **Appendix B – BIRO Database Manager source code**

---

## **Appendix C – BIROBox source code**

---