



EUBIROD
European Best Information
through Regional Outcomes in Diabetes



Executive
Agency for
Health and
Consumers

Work P7 Customized Toolbox

Deliverable 7.1

Version 2.1

AUGUST 2010

This report is the Deliverable D7 of “WP7: Customized Toolbox” of the European project “European Best Information through Regional Outcomes in Diabetes” (EUBIROD), co-funded by DG-SANCO, European Commission, 2008 (G.A. 2007115)

Joint production of the EUBIROD Consortium:

*Adelaide and Meath Hospital, Dublin, Ireland
Centre Hospitalier de Luxembourg, Luxembourg
Dasman Centre for Research and Treatment of Diabetes, Kuwait
Dutch Institute for Healthcare Improvement, Netherlands
Havelhöhe, Berlin, Germany
Hillerød University Hospital, Hillerød, Denmark
IMABIS Foundation, Malaga, Spain
International Diabetes Federation, Brussels, Belgium
Scientific Institute of Public Health WIV, Brussels, Belgium
Joanneum Research, Graz, Austria
Medical University of Silesia, Katowice, Poland
Ministry of Health, Cyprus
NEPI Foundation, Malmö, Sweden
Paulescu Institute, Bucharest, Romania
Sereatrix snc, Italy
NOKLUS, Bergen, Norway
University of Dundee, Scotland
University of Malta, Malta
University of Perugia, Italy
University of Debrecen, Debrecen, Hungary
University Children’s Hospital, Ljubljana, Slovenia
Vuk Vrhovac University Clinic for Diabetes, Zagreb, Croatia*

WP Leader

JOANNEUM RESEARCH, Austria

Compiled for JOANNEUM RESEARCH by:

Peter Beck, Clinical Expert
Philipp Perner, Software Engineer

Citation

P.Beck, P.Perner on behalf of the EUBIROD Consortium, EUBIROD Customized Toolbox, EUBIROD Consortium, 2010

Address for correspondence

JOANNEUM RESEARCH Forschungsgesellschaft mbH
HEALTH - Institute for Biomedicine and Health Sciences
Elisabethstrasse 11a, 8010 Graz – Austria
Phone +43 316 876-2131
Fax +43 316 879-2130
Email: health@joanneum.at

Project-Website

www.eubirod.eu

Contents

| | | |
|-----|---|----|
| 1. | Document Change History | 5 |
| 2. | Introduction..... | 5 |
| 3. | Objectives..... | 6 |
| 4. | Customizable Toolbox Content and Evolution..... | 6 |
| 4.1 | Support installation and execution of BIRO software | 6 |
| 4.2 | Support data transformation and manipulation..... | 7 |
| 4.3 | Support the BIRO software development process..... | 7 |
| 5. | Supporting BIRO Software Development..... | 8 |
| 5.1 | Theoretical Background..... | 8 |
| 5.2 | Results..... | 13 |
| 6. | BIRO Software Installation on Microsoft Windows | 15 |
| 6.1 | Materials and Methods | 15 |
| 6.2 | Results..... | 16 |
| 7. | Running BIRO Software on Linux..... | 20 |
| 7.1 | Background | 20 |
| 7.2 | Results..... | 23 |
| 8. | Executing BIROX on a Virtual PC | 30 |
| 8.1 | Background | 30 |
| 8.2 | Results..... | 31 |
| 9. | Extended Data Transformation..... | 32 |
| 9.1 | Materials and Methods | 32 |
| 9.2 | Results..... | 36 |
| 10. | Appendix A | 42 |
| | Download and Installation of Oracle VirtualBox | 42 |
| | Download of the Virtual Disc Image of BIRO | 45 |
| | Usage of BIROX in Oracle VirtualBox | 45 |
| | Configuration of the Exchange Directory..... | 54 |
| | Additional Information..... | 55 |
| 11. | Appendix B | 57 |
| | Maven's Default Lifecycle Phases..... | 57 |
| 12. | Appendix C | 59 |
| | Technical Infrastructure Questionnaire | 59 |
| 13. | Appendix D | 63 |
| | Technical Infrastructure Summary | 63 |
| 14. | References | 68 |

Figures

| | |
|--|----|
| Figure 1 – Software Development Lifecycle using iterative feedback loops..... | 9 |
| Figure 2 – Interaction of Lifecycles, Phases, Plugins and Goals in Maven | 11 |
| Figure 3 – Software Deployment Process..... | 12 |
| Figure 4 – BIRO software’s multi-module project structure using inheritance of POMs | 14 |
| Figure 5 – Start screen of the BIRO-Installer containing Copyright Information (Step 1) | 17 |
| Figure 6 – (Initial) Installation Options of the BIRO system (Step 2)..... | 17 |
| Figure 7 – Dialog to choose the installation directory (Step 3) | 18 |
| Figure 8 – Start Menu Folder for Windows (Step4)..... | 18 |
| Figure 9 – Installation of the BIROBox (Step 5)..... | 18 |
| Figure 10 – Dialog to confirm the location of R and Java on the system (Step 6) | 19 |
| Figure 11 – Closing dialog of the installation routine (Step 7) | 19 |
| Figure 12 – Use Case for installing new software packages | 22 |
| Figure 13 – Build structure of the BIRO System..... | 23 |
| Figure 14 – Configuration-file for Aptitude to manage the software repositories | 26 |
| Figure 15 – Configuration wizard for Aptitude to manage software repositories | 27 |
| Figure 16 – Using aptitude to search the software repositories for the term “biro” | 27 |
| Figure 17 – Using aptitude to install the BIRO software-packages | 28 |
| Figure 18 – Update notification of installed software in Gnome Desktop | 28 |
| Figure 19 - DVD label of the BIROX distribution..... | 29 |
| Figure 20 – Mapping between data source fields and BIRO data items within the BIROBox. The BIROAdaptor uses the information of the mapping for the transformation to the format of the BIRO Common-Dataset. | 32 |
| Figure 21 - Excerpt of a SQL-query including transformations to create a database view | 33 |
| Figure 22 – ETL workflow in Kettle using jobs and transformations..... | 35 |
| Figure 23 – example of a transformation in Kettle using the Pentaho Spoon workflow-designer..... | 35 |
| Figure 24 - PDI-job for the BIRO system | 36 |
| Figure 25 - Example of a country-specific transformation in PDI | 37 |
| Figure 26 - “Modified Java Script Value”-dialog of PDI’s Spoon to transform values of data fields really fast on a script basis. | 37 |
| Figure 27 - Dialog which provides functionality to select, alter, remove or modify existing data fields | 38 |
| Figure 28 - BIRO-Validator transformation. The dataset is split into correct and incorrect data fields..... | 38 |
| Figure 29 - Data validation of the BIRO data item 012: HEIGHT..... | 39 |
| Figure 30 - Overall process of data import within the BIROBox | 40 |
| Figure 31 - Merge-Table source configuration dialog using a csv-file and a customized configuration designed in PDI..... | 41 |

1.Document Change History

| Version | Date | Author | Reason for Update |
|---------|---------------------------------|----------------------------|--|
| 0.1 | June 2010 | Philipp Perner | Initial Draft |
| 1.0 | July 2010 | Philipp Perner | Update for the first version |
| 1.1 | 15 th August 2010 | Philipp Perner | Revision of the document |
| 1.2 | 20 th August 2010 | Peter Beck | Review of the document |
| 2.0 | 29 th August 2010 | Philipp Perner | Update of the document |
| 2.1 | 31 st August 2010 | Philipp Perner | Review and preliminary submission |
| 2.2 | 27 th September 2010 | Peter Beck | Review |
| 2.3 | 22 nd October 2010 | Peter Beck | Review, streamline sequence of chapters |
| 2.4 | 2 nd October 2010 | Philipp Perner | Review and update of technical descriptions |
| 2.5 | 3 rd October 2010 | Philipp Perner, Peter Beck | Review and update |
| 2.6 | 2 nd November 2010 | Philipp Perner | Modifications after feedback from Fabrizio Carinci, Biro Box Screenshot update |
| 2.7 | 3 rd November 2010 | Peter Beck | Final review |

2.Introduction

EUBIROD aims to implement a sustainable European Diabetes Register through the coordination of existing national/regional frameworks and the systematic use of the BIROⁱ technology. The system will fulfil the Conclusions of the EU Council for systematic data collection and monitoring of diabetes complications and health outcomes across Europeⁱⁱ.

EUBIROD targets the sustainability of complex systems of health indicators requiring continuous update and regular maintenance. The project proposes an action to implement, extend, and customize the application of the BIRO technology in at least 20 States, including EU Member States, Acceding/Candidate Countries, and EFTA Countries.

Participants will be connected through a system that will safely collect aggregated data and produce systematic EU reports of diabetes indicators, which will be used to develop recommendations for policy makers.

The EUBIROD Consortium includes all BIRO partners (N=7), new partners (N=12) coming from all over Europe and two collaborating institutions.

To extend the BIRO system to the EUBIROD Consortium the dissemination of the BIRO approach was required as preliminary step in addition to effective training for new partners on the BIRO solution and preparation for the frequent and unattended usage of the technology by all partners.

This report aims to summarize the activities done for the work package seven "Customized Toolbox" and the achieved results.

3.Objectives

The aim of WP 7 - "Customized Toolbox" was to create a distributable piece of software for the EUBIROD partners to execute and use the BIRO system. Moreover the possibilities to process data and contribute results to BIRO should be improved to support more users in different environments. While the main aim was to make usage of the software as easy as possible for users, this lead to more complex requirements from the technical perspective and an extended functional range of data integration.

The primary task was to create a solution to **facilitate the usage of the software** for project partners and prospective BIRO system participants. In general the software aims to facilitate preparation, handling and import of data. It supports submission of aggregated regional data and allows the regional users to benefit from BIRO results.

As regional data exists in many different formats, there is always a possibility for exceptional cases that can not be handled by the BIROBox (Software developed within BIRO for the handling of data). So a second task in this work package was to provide and integrate an additional tool to **extend the functionality of the data-integration and data-transformation.**

4.Customizable Toolbox Content and Evolution

4.1 Support installation and execution of BIRO software

- The first aid to set up the BIRO software was a HOW-TO guiding the user through the set up with many manual steps. In a training session in Kuwait in May 2009 the training phase dealing with the setup was considered the most difficult aspect of the training session (see Deliverable 4.1 – Report on Trainingⁱⁱⁱ).
- A decision was made to create an Installer to automatically set up the BIRO software and all additional software packages on a *Microsoft Windows* operating system.
- During the meeting in Rome in November 2009 the Windows installer was comprehensively tested by all EUBIROD partners for the first time. Not in all cases the software could be successfully installed by all partners due to several reasons such as different operating system versions, different pre-installed software and resulting version conflicts etc.
- During several subsequent brainstorming sessions among the EUBIROD consortium solutions were discussed how a BIRO system could be created even more "out of the box" to be more independent from operating system and pre-installed software on the user's PC. The selected way to achieve this aim was the creation of a standardized software environment based on the Linux operating system, containing the necessary software for running the BIRO system.

As a consequence the following possibilities to set up and execute BIRO software are available to date:

- A **package management infrastructure for Linux** to distribute the BIRO software
- The **BIRO Linux distribution (available on DVD or as virtual appliance)**: A standardized environment containing the BIRO system and the additional software programs and components to work out of the box without interference with the user's operating system and other installed software.
- The **BIRO Windows Installer**: Automatic installation of BIRO software and all required programs and components on Microsoft Windows Operating Systems.

4.2 Support data transformation and manipulation

During the first tests with user data in the project it became clear that regional data exists in many different formats putting complex requirements on connectors and data transformation. To assess the requirements in more detail and to estimate the variety of data-sources and data-formats, a questionnaire was sent out to participating partners. The complete questionnaire is shown in Appendix C

Technical Infrastructure Questionnaire

. The questionnaire was returned by a total of 14 partners (N=20). A summary of the information retrieved is shown in Appendix D. The summary shows a general heterogeneity of data-sources and data-formats among the countries.

It was observed that the BIROAdapter will have to deal with a big diversity of values for single data fields, which could result in a relatively high effort to develop a strategy for data integration into the BIRO system. It was decided that to a certain extent data-integration and data-transformation and the mapping of local datasets to the BIRO Common-Dataset definitions should be implemented by the BIROAdapter.

Special cases and requirements not covered by the BIROAdapter should be covered by an additional set of tools as part of the Customized Toolbox. So the aim of this work package was to extend the functionality available for data-integration and data-transformation in BIRO.

To achieve this aim, a possibility to execute **customized ETL (extraction, transformation, loading) processes** for individual data sets was integrated into the BIROBox to seamlessly be available for the users.

4.3 Support the BIRO software development process

BIRO software development was started in several stand-alone software projects. Several dependencies existed between these software-projects. Furthermore, external libraries are used by almost all of them. Soon it became very complicated to build and update the system, so that the requirement for a unified project structure and the ability to manage dependencies between software components and external libraries became evident.

This could be reached by porting and merging the existing BIRO software projects to one project and reorganising build process and project management according to Apache Maven¹.

5.Supporting BIRO Software Development

5.1 Theoretical Background

Activities, which reach from a proper developed software to a product that can be delivered to consumers as final product are very comprehensive. Many different aspects of the build-environment, the build-structure, the software release-strategy as well as the software's bug-fixes, change-management and its corresponding release-cycles have to be taken into account. For a better understanding of the corresponding activities in the software-development- and deployment-process, a brief overview will be given in this chapter.

5.1.1 Development Process

Software products, as stated in the waterfall model^{iv,v}, are the result of an iterative software development process where requirements have been analyzed and specified and the software has been designed, developed and properly validated using tests. This chapter describes an efficient possibility to create deployable software using Apache Maven.

Many models for higher sophisticated processes have been published since then. (Ruparelia, 2010)^{vi} provides a brief overview of existing models like the incremental model, V-model, spiral model and the rational unified process model.

Subsequent phases in the waterfall model are the deployment-phase including the maintenance of software. These phases will be covered in chapter 5.1.2 Software Deployment Process. Figure 1 shows the waterfall model and its phases.

¹ Apache Maven. 13th of August 2010, from maven.apache.org

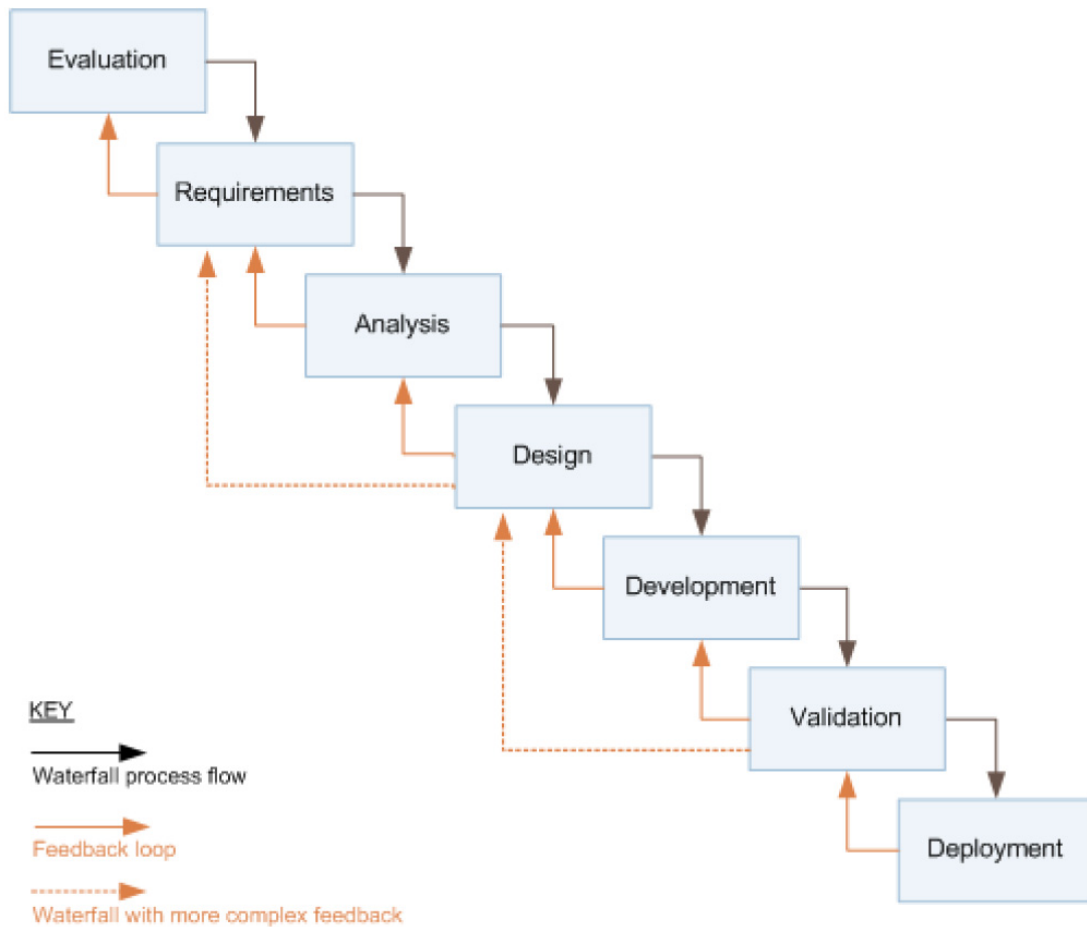


Figure 1 – Software Development Lifecycle using iterative feedback loops^{vi}

Apache Maven is a tool used in Java to facilitate many aspects of the software development process. Although Maven was initially started as build tool which should replace Apache Ant², its functional variety in version two is now far beyond Apache Ant and can be best described as software project management tool. Maven's main goal is to help developers comprehending the complete state of development effort anytime during the software project. In order to obtain this goal, there are several objectives declared by the project team:

- Making the build process easy: Details from underlying mechanisms to build software are shielded from the user
- Providing a uniform build system: Maven's concept of inheritable project object models (POMs) and a set of plugins, which are shared among all maven projects, allows users to familiarize themselves with the builds, which are similar for all maven projects.
- Providing quality project information: Useful project information can be extracted from the POM-file or the source code like changelog, mailing lists, library dependency lists, unit test reporting including code coverage and cross referenced sources.
- Providing guidelines for best practices development: Maven aims to gather current principles of best practices in software development, called maven archetypes, like:
 - Running tests as part of the normal build cycle

² Apache Ant. 13th of August 2010, from ant.apache.org

- Providing assistance in project workflow like release management and issue tracking
- Defining the project's directory structure according to existing guidelines for currently more than 250 different types of Java projects, like EJB³ in Java EE 6, Web-projects⁴ in Java EE 6, Spring Web-Services⁵ or many different AppFuse⁶ based projects.
- Allowing transparent migration to new features: Every time maven is called for the project's build, maven checks for itself for updates so the clients make advantage of new features.

To meet these objectives Maven provides features like fast project setup using archetypes, plugin- and dependency-usage among all projects, superior dependency management including automatic updating and resolving of transitive dependencies⁷.

When it comes to software build, maven provides features to build any number of projects into predefined types such as JAR, WAR and EAR. In terms of software-quality and internal or external documentation maven is capable of generating project-documentation or maven project-sites. During the deployment phase, maven can publish built software-pieces into distribution-locations like the locale maven-repository or any other location so the software can be used in other projects. Maven is also able to release the software as binaries (JARs), including their dependencies, or as a source distribution for further development or to meet the requirements of license-conditions. The idea of dependency-management, which is the backbone of maven, has the advantage that these dependencies, i.e. java libraries, are stored in a local repository so they can be reused across all projects which encourages communication between projects to ensure backward compatibility of the software.

Maven itself is a plugin execution framework. Every goal's functionality in maven is provided by either a core plugin or another plugin. An overview of existing plugins can be seen on the Maven Plugin-Site⁸.

The whole definition of a Maven project, the project's usage of plugins and which tasks should be performed on which point in time is defined in the POM-file (*pom.xml*). The POM-file contains the whole identity of a project:

- What does a project contain?
- What type of packaging needs the project?
- Does the project have a parent POM?
- What are the dependencies?

³ Enterprise Java Beans 3.0 Final Release, 17th of August 2010, from <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>

⁴ Oracle Web Tier, 17th of August 2010, from <http://www.oracle.com/technetwork/java/webtier/overview/index.html>

⁵ Spring Web Services, 17th of August 2010, from <http://static.springsource.org/spring-ws/sites/1.5/>

⁶ AppFuse2, 17th of August 2010, from <http://appfuse.org>

⁷ A transitive dependency is a dependency of another dependency, which results in a dependency tree. The software developer has to define only the dependencies he directly uses in his source code or he uses during runtime.

⁸ Maven - Available Plugins. 18th of August 2010, from maven.apache.org/plugins

POM-files are declarative descriptions of projects so Maven can understand what it should look for when the project is built. For further information about the Project Object Model in Maven please take a look at Chapter 3 of (O'Brien, 2010)^{vii}.

In order to allow maven to act upon the descriptions made in the POM-file, Maven uses goals packaged in Maven plugins which are tied to phases in a lifecycle. An overview of this interaction is given in Figure 2.

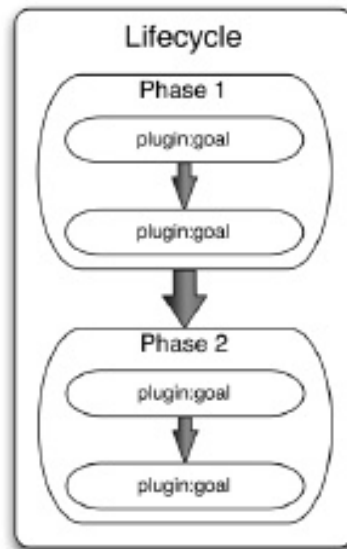


Figure 2 – Interaction of Lifecycles, Phases, Plugins and Goals in Maven

Built-in lifecycles in Maven are the clean-lifecycle and the default-lifecycle. The clean-lifecycle contains the phases pre-clean, clean and post-clean. The clean's plugin clean goal (clean:clean), which is bound to the clean phase, deletes the output folder of a build by deleting the build directory.

A complete reference of the default-lifecycle is listed in Appendix B. As mentioned above, Maven is also capable of building a project-site. This functionality is provided by the site lifecycle.

Maven is able to create packages of a project in different forms like JAR, POM, EAR, Maven Plugin, EJB, WAR or many more. This functionality is called package-specific lifecycle, because lifecycles have bound default phases with the same name to different goals. So the package-element in the pom.xml serves as a switch for different functionality. For an example of how the packaging affects the build one can consider two projects, one with JAR packaging and the other one with EJB packaging. Whereas the package phase of the first project calls the goal jar:jar, the second project's package phase will call the goal ejb:ejb.

A proper software development process, where efficient tools to manage dependencies and to test and build the source-code are used, is the foundation for a proper software-product.

5.1.2 Software Deployment Process

Software deployment can be seen as a process which comprises many sophisticated interrelated tasks like the efficient release, installation and maintenance (activation and update) of software as well as its de-installation, adaption, de-activation and de-release on a large amount of clients. An overview of a complete deployment process can be seen in Figure 3. Although it contains typical activities in

deployment processes, the effort being made for each activity has to be regarded specifically for each software product and its deployment process. A brief description of the activities and their subtasks is given below.^{viii}

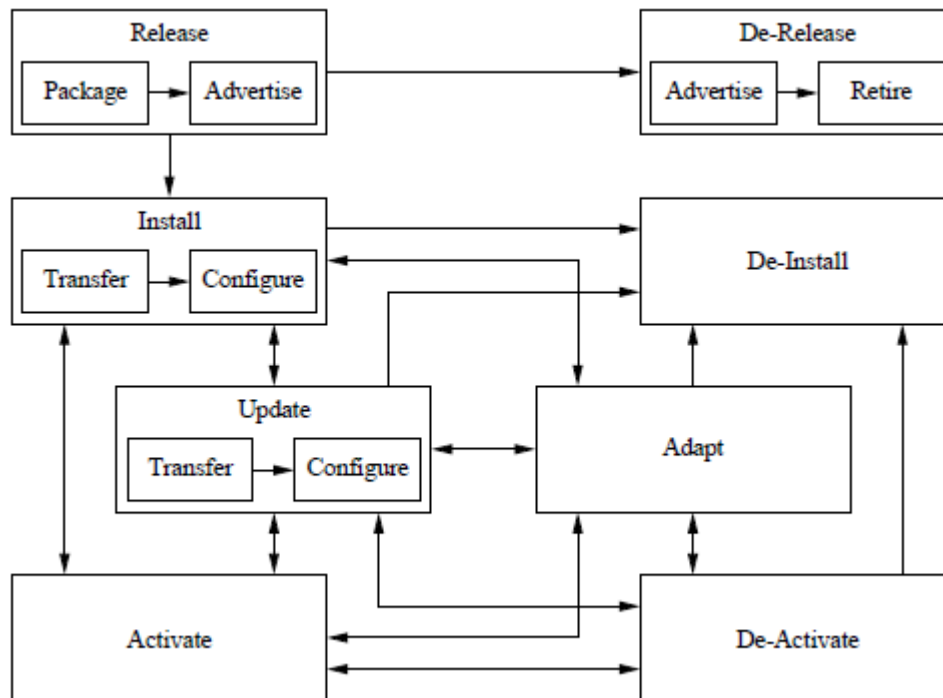


Figure 3 – Software Deployment Process^{viii}

Release: The release can be seen as interface between the development process and the deployment process. It contains the activities from assembling software from the developer's environment, i.e. the build structure to test software in the development process, to a structure where it can be used on the consumer environment. The release activity has to make correct assumptions of the user environment and provides resources for the software to run correctly on the consumer site. It contains the *packaging* subtask, which means to build a transferrable unit of the software containing its system components, a description of the system, the deployment procedures, its requirements and dependencies and the subtask *advertising*, which provides the users with appropriate information of the software.

Install: This task is the initial installation of the software on the client side. It deals with the users' environment and the proper adoption of the system without causing side effects, inconsistencies or harm on the client's system. Its subtask is the correct transfer and configuration of the software.

Activate: Activation means any subtask of the software until being executable on the client side. This entails the correct start-up of the depending software (databases, servers, system daemons...) as well as establishing some form of command to start the program (start menu entry, desktop icon, executable file in the path, ...).

De-Activate: De-activation is the opposite of activation and restores the system's state before activation. This task is required often before other deployment activities like updating or de-installing.

Update: An update is a special version of the installation process. It often requires de-activation, installation of a new version and re-activation. An update can often rely on existing configurations, which are done in the installation phase. New configuration parameters, which could be necessary for changed features, have to be considered. Like the installation, the update needs to transfer and configure the software package.

Adept: This activity, like the update activity, involves modifying previously installed software. In contrast to the update activity, which is triggered remotely, adept activities describe measures being taken so the software remains executable, when local changes in the environment are detected.

De-Installation: If a software package is no longer needed on the client, the best case is to restore the system's state as it was before installing the software. During removal, existing system parameters and properties have probably to be reconfigured instead of removed so the system will still work. Deinstallation is a critical issue, when dependencies amongst other software packages may arise, which have to be taken into account.

De-Release: When no further development for software is given, the package has to be marked as obsolete. Beneath withdrawing the support, the withdrawal has to be advertised to all known customers and the hosts of the software-packages.

5.2 Results

The BIRO software for the client side, i.e. BIROCommunicationSoftware, BIROAdaptor, BIROStatisticalEngine and the BIRODatabaseManager, was split into different subprojects at the beginning of the BIRO project and therefore developed by different consortium members of the project. At the end of the BIRO project, the idea of a GUI⁹, called BIROBox, was introduced to the project consortium, which should connect the tools mentioned above to one comprehensive tool. The BIROBox should act as a container covering the interfaces of the tools among each other and provide the trigger for the complete process of the BIRO system.

The BIROBox, and its containing subprojects, was developed using the Netbeans¹⁰ IDE¹¹. The software project was technically initiated using a Netbeans-proprietary project structure. On the one hand, this technique has the advantage that software development can start immediately without much organisational overhead. On the other hand the proprietary project structure does not provide an efficient project structure which facilitates building, testing, deploying and maintaining the software. This has the disadvantage that customizable builds of multiple projects, like in the BIRO project (BIROAdaptor, BIROBox, BIRODatabaseManager and BIROCommunicationSoftware), are difficult to implement, because the projects strongly interact with each other.

For this purpose, a new project structure was introduced based upon Apache Maven.

⁹ GUI...Graphical User Interface

¹⁰ Netbeans IDE. 23rd of August 2010, from www.netbeans.org

¹¹ IDE... Integrated Development Environment

Apache Maven provides Java archetypes for the types of Java-projects created so far, e.g. "Swing Application Framework (JSR 296)"-Archetype and "Hibernate Archive"-Archetype. Furthermore Maven is fully integrated into the Netbeans IDE, so the developer can use the full functionality of Maven for the whole software development process within the IDE.

The project structure was designed according to the best-practice model of a multi-module project using inheritance referring to Chapter 3.6.2 of (O'Brien, 2010)vii. Figure 4 shows this project structure:

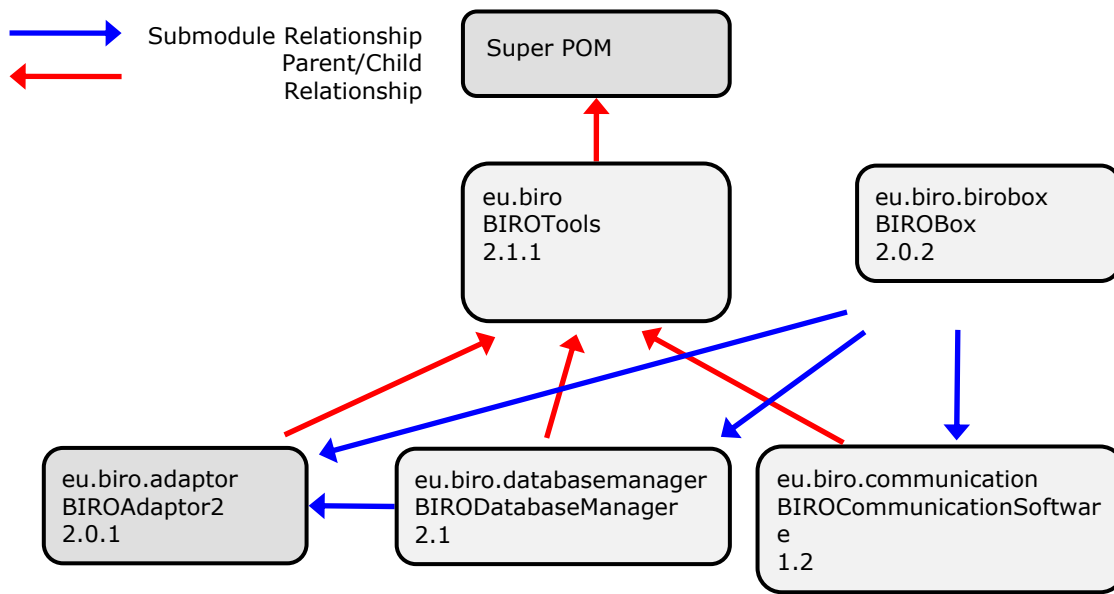


Figure 4 – BIRO software’s multi-module project structure using inheritance of POMs

When the project has been checked out via the BIRO SVN, the BIRO software can be build using either Netbeans IDE to open and build the project or using the command-line at the location of the Super POM using the command:

When using the command

```
$ mvn package
```

Apache Maven builds the BIRO software in two different formats:

- As assemblies using the *maven-assembly-plugin*, which packs the SVN projects into binary-, source- or maven-project-packages for the purpose of software-distribution for interesting parties beyond the project team.
- As executable software, that can be subsequently used to create either Debian-packages or an installation-routine for Microsoft Windows.

Due to Apache Maven’s dependency management the source-code of the BIRO system (10MB) is five times smaller then the actual build (50MB) and the build, compared to the former build-environment using the proprietary project-structure of Netbeans is seven times smaller (350MB). The efficiency of the new solution has

tremendously improved. The overall build process, if not run for the first time¹², takes less than a minute compared to nearly five minutes using the former build-environment.

6. BIRO Software Installation on Microsoft Windows

6.1 Materials and Methods

Building Software-packages and deploying them for Microsoft Windows is even more challenging as the deployment on Linux. Microsoft Windows doesn't provide a similar functionality like the technology of package management systems in Linux. Every piece of software, a user wants to install, has either to be downloaded from a website in the internet or provided via a software medium like a USB-stick, a CD or anything similar. For this purpose it's convenient for a user to deal only with a single executable file, which takes care of the proper (de-)installation, (de-)activation, adaption, and update. Many tools and techniques have been introduced to the software-market to provide assistance in the field of installation-routines in Windows. A commercial Product to setup software on Windows is *Install Shield*¹³. Free alternative installing tools are *NSIS*¹⁴ (Nullsoft Scriptable Install System), *InnoSetup*¹⁵ and *WiX*¹⁶.

Some of these tools make use of the Windows built in system component *Windows Installer*¹⁷ which is a product provided by Microsoft used for installation, maintenance and removal of software. The installer can be recognized by its default file extension *msi*. Windows Installer works directly with Windows system components, so correct synchronization and therefore consistency of new software packages with already installed software and the operating system itself can be maximized. Windows Installers need a separate authoring tool to create *msi*-files, which exist as free or purchasable software.

A short overview of each installation tool will be given below.

Install Shield is a proprietary installation solution for Microsoft Windows using Windows Installer. Install Shield's features vary greatly among the Express-, Professional- and Premier-Editions. Whereas the Express-Edition is a wizard-based assistant for creating Windows-Installer setup routines, the professional edition provides support for creating customized GUI-dialogs, setup-scripts and the reuse of formerly created setup-solutions. The Premier Edition uses Install-Shield merely as compiler, so the authoring of the setup-routine can be created with other tools than Install-Shield.

¹² Apache Maven needs to download dependencies and plugins if it is running for the first time. This can take for the BIRO software, depending on the connection to the internet, up to 20 minutes.

¹³ Install Shield – MSI Windows Installer and Install Script Software Installations. 20th of August 2010, from <http://www.flexerasoftware.com/products/installshield.htm>

¹⁴ NSIS – Nullsoft Scriptable Install System. 20th of August 2010, from nsis.sourceforge.net

¹⁵ Inno Setup. 20th of August 2010, from www.jrsoftware.org/isinfo.php

¹⁶ WiX Toolset. 20th of August 2010, from wix.codeplex.com

¹⁷ Windows Installer. 20th of August 2010, from <http://msdn.microsoft.com/en-us/library/cc185688.aspx>

NSIS is a script-based program to create a wizard-based installation routine for Microsoft Windows and Linux. Due to its free and open source license, NSIS is widely used, e.g. Winamp, Firefox, VLC, IrfanView and OpenOffice. Many plugins and scripts are already available for installing, updating, uninstalling extracting files or communicating with the operating system through the Windows API. NSIS can read from and write into the Windows registry. It was created to be small, fast and efficient so a full installation routine's logic is stored in only 34 KB of overhead. NSIS scripts can be created using third party software like NSIS plugin for Eclipse¹⁸ or HM NIS Edit¹⁹.

Inno Setup is a program to create script based installation routines. Inno Setup comes with an editor supporting syntax highlighting. Using IStool²⁰ developers can create scripts using a graphical user interface. Third party software and plugins exist for Inno Setup, due to its open source-code, to enhance the functionality of Inno Setup. Its feature list is comparable to the features of NSIS.

WiX is a free software toolset for Windows Installer to build *msi*-files from declarative *xml*-Files. It was released by Microsoft as first open-source licensed product in 2004. WiX can be used as command line tool using WiX tools or MSBuild. Furthermore, Microsoft released a Visual Studio plugin for VS 2005, VS2008 and VS2010. WiX itself consists of a collection of tools to compile (Candle), load libraries (LIT), create software patches (Pyro) and create *msi*-files (Light). The bootstrapping functionality for software and its installation (Burn) is about to be released in WiX 3.6.

6.2 Results

The installation routine of the BIRO system for Windows was created using the open source script-driven installation system NSIS²¹. This tool allows to read and to write *Microsoft Windows'* registry entries, to set system settings, system variables and to add a routine to uninstall the software.

The installer is designed as wizard containing every step to get the BIRO system running. The steps needed to setup the BIRO system using the installer will be described in detail below.

¹⁸ EclipseNSIS. 16th of August 2010, eclipsensis.sourceforge.net

¹⁹ HM NIS Edit – a free NSIS editor/IDE. 16th of August 2010, hmne.sourceforge.net

²⁰ IStool. 20th of August 2010, www.istool.org

²¹ Nullsoft scriptable install system. 14th of July 2010, from <http://nsis.sourceforge.net>

Step 1 contains the copyright description of the Software developed during the project. As agreed in the project every single piece of software will be distributed under the GNU General Public License v.2 or later. See Figure 5.

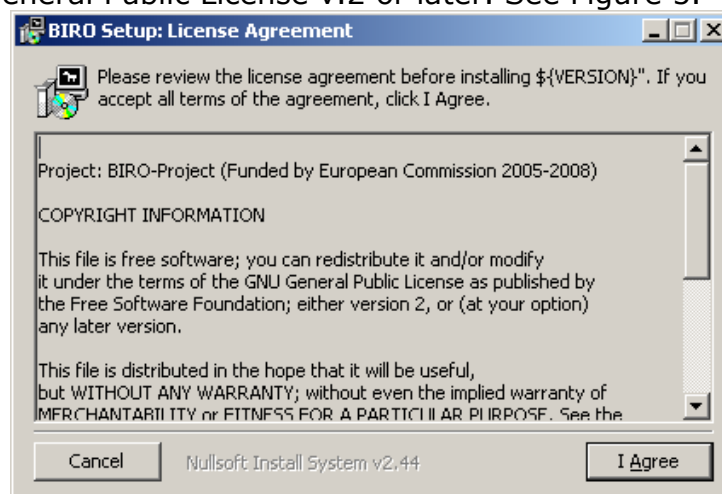


Figure 5 – Start screen of the BIRO-Installer containing Copyright Information (Step 1)

Step 2 lists the available software products needed by the BIRO system to be installed, i.e. BIROBox, Database (Postgres 8.3), Statistical Software (R 2.8.0), Report Tool for creating PDFs (Miktex 2.7.0), Java (JDK 1.6) and the Crypto Extension for Java as needed by the *BIROCommunicationSoftware*. The installer detects through Microsoft Windows' registry which piece of software is already available on the system and based on this information the checkboxes are initially marked as whether checked or unchecked. See Figure 6.

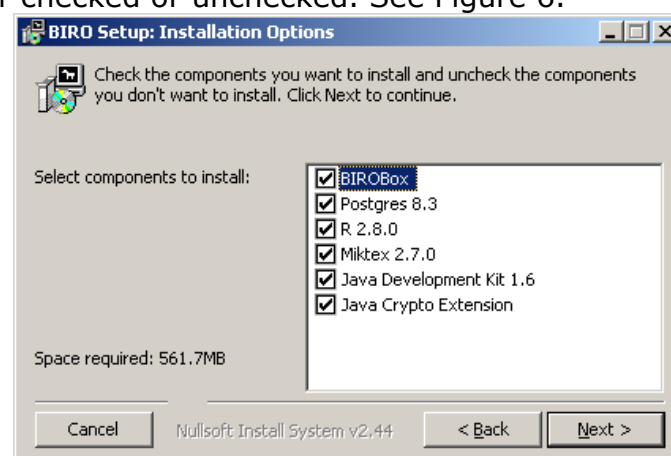


Figure 6 – (Initial) Installation Options of the BIRO system (Step 2)

In **Step 3**, the user can choose the installation directory of the BIRO system. In **Step 4**, it's possible to choose the name of the Start Menu folder where the shortcuts to the BIROBox should be placed. See Figure 7 and Figure 8.

Step 5 performs the installation of the BIROBox (Figure 9)

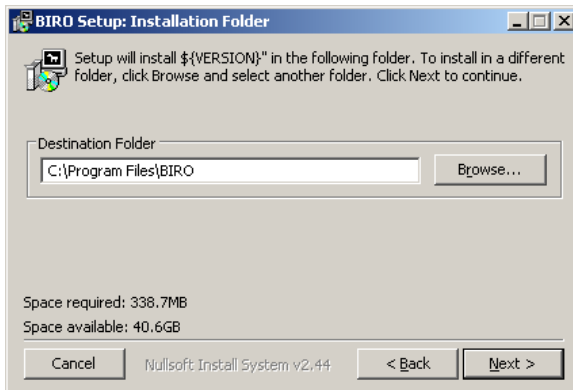


Figure 7 – Dialog to choose the installation directory (Step 3)

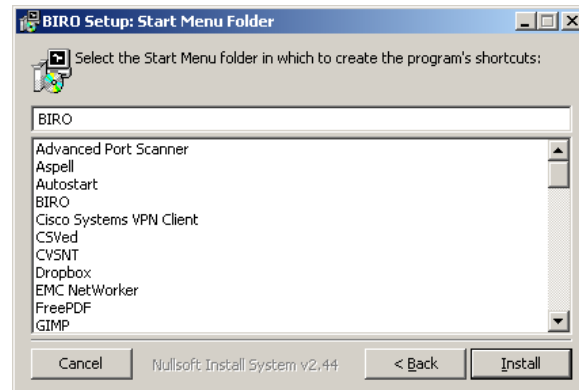


Figure 8 – Start Menu Folder for Windows (Step4)

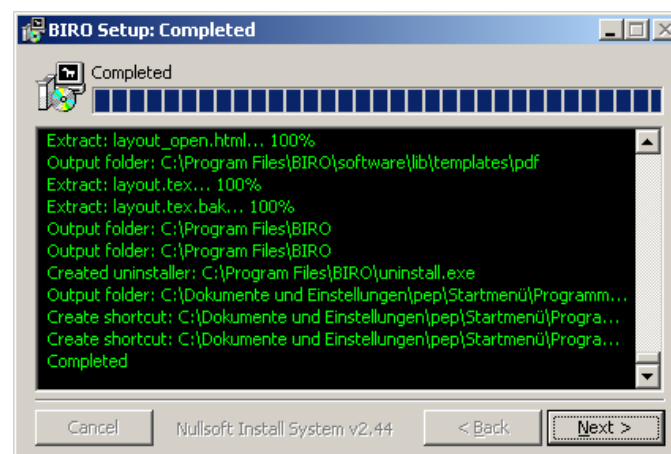


Figure 9 – Installation of the BIROBox (Step 5)

After the "Next" button is clicked in **Step 5**, the installer executes every single installation program of the additional software, which had been marked as checked in Step 2. These programs are part of the BIRO-Installer and the user does not need any additional downloads. When the additional software is either already available on the system or has finished with the installation, the BIRO-Installer continues with Step 6.

Step 6 is required to set system variables for Windows according to the installation paths of the programs *R* and *Java*. The installer tries to read this information from the registry and initially presents these suggestions in the textboxes with an option to set alternative paths using the "Browse" button. See Figure 10.

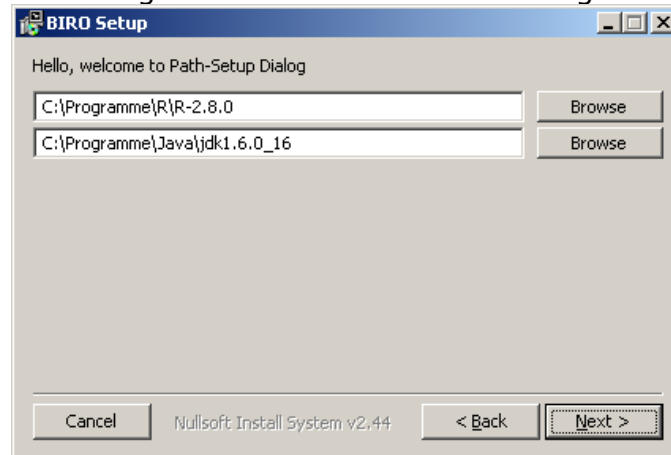


Figure 10 – Dialog to confirm the location of R and Java on the system (Step 6)

In the last step, **Step 7**, it is recommended to restart the system before using the BIRO system for the first time. See Figure 11.

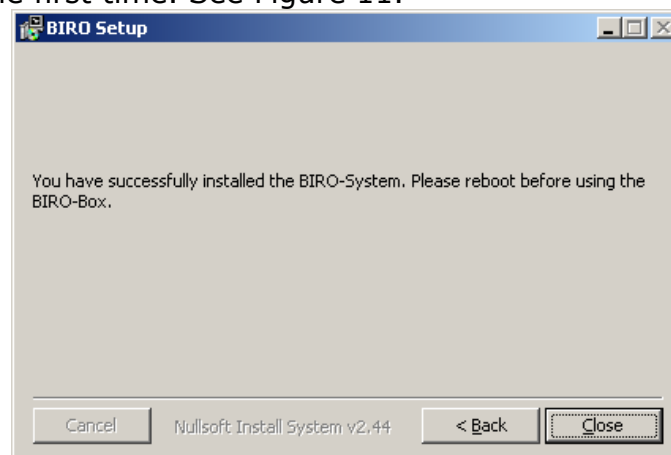


Figure 11 – Closing dialog of the installation routine (Step 7)

In comparison to the "*BIROBox Setup Guide*" in Appendix C of *Deliverable 4.1 – Report on Training*ⁱⁱⁱ, the BIROInstaller saves a tremendous amount of time until the system is set up and executable. The BIROInstaller, containing BIROBox v1.0.4, was first released in June 2009. Until the meeting in November 2009 in Rome, several software-improvements and bugfixes were performed, so BIROBox v.1.0.6 could be presented and tested by each partner during the "*Hands-On*"-Session in Rome.

Several limitations have been revealed during testing this installer and running the BIROBox:

- Many partners use notebooks provided by their companies or hospitals. The operating systems do often have permission constraints for their users. That's why hardly any software can be installed due to security reasons.
- Because of the secure software- and data-environment in some companies or hospitals, it's not allowed to install personal software on the computers at all.

- Some partners have already installed different versions of the additional software on their computers, which lead to version conflicts during the runtime of the BIROBox.
- The BIRO system is a growing piece of software, which has to be improved as well as bug-fixed in the future. For this purpose the Windows installation routine has to provide the possibility to update the software easily. This feature is difficult to implement in NSIS. The installer currently does not support update functionality.

Due to these limitations the creation of the Windows installer can be considered part of the learning curve during this project. Currently it is recommended to execute BIRO software bundled in the BIROX distribution described in chapter 8 Executing BIROX on a Virtual PC.

7. Running BIRO Software on Linux

To make the BIRO software executable on Linux it is necessary to perform some steps to meet the special requirements for Linux. These steps will be described in the following chapters.

7.1 Background

Linux or GNU/Linux is a UNIX-like operating system. Although it was usually designed as free and open source alternative to UNIX, which was mainly used on server environments, its wide variety of distributions makes its application in recent years also common on desktop environments. Currently many different distributions exist, e.g. Debian²², Fedora²³, Knoppix²⁴, openSuse²⁵, RedHat²⁶, Suse²⁷ or Ubuntu²⁸. Especially Ubuntu has made great effort in the area of desktop usage in the recent years. Due to the great variety and diversity of software packages in its software repository as well as its main focus on end-users instead of system administrators, Ubuntu was regarded as the most suitable for the application in the EUBIROD project.

Ubuntu is a Linux based on the Debian distribution. It uses many features which are users known from the Debian system. An advantage of Ubuntu is that developers are focussed on supporting new hardware. Next to the official packages available for Ubuntu, many additional repositories exist, which contain different free and non-free software to enhance the availability of software-products. Ubuntu makes use of

²² Debian – The Universal Operating System. 11th of August 2010, from www.debian.org

²³ Fedora – Project. 11th of August 2010, from www.fedoraproject.org

²⁴ KNOPPIX – Live Linux Filesystem on CD. 11th of August 2010, from www.knopper.net

²⁵ openSuse.org. 11th of August 2010, from www.opensuse.org

²⁶ Redhat.com | The World's Open Source Leader. 11th of August 2010, from www.redhat.com

²⁷ Linux OS| SUSE Linux Enterprise. 11th of August 2010, from www.novell.com/linux

²⁸ Ubuntu homepage | Ubuntu. 11th of August 2010, from www.ubuntu.com

the widespread Advanced Packaging Tool²⁹ (APT) as Package Management System (PMS), which is based on the usage of Debian-packages.

7.1.1 Building Software Packages for Debian

Debian uses packages called Debian-packages for the distribution of software. A Debian-package can be regarded as container, consisting of the software itself and metadata containing instructions for the package management system. The metadata stores the information how the software interacts with the client system, i.e. install location, revision and dependencies to other software-packages. Building Debian-packages requires exactly this information, so this can be stored in the metadata. Furthermore the location of the source-files and optional instructions before and after the installation can be configured.

Before building the packages, there are a number of files which have to be edited in order to customize the behaviour of the software. A complete reference, which files should be edited to build Debian-packages, is listed in (Rodin, 2010)^{ix}.

Some of the main parts of the configuration are:

- File *control* → defines available packages and dependencies among other packages, e.g. third party software
- File *changelog* → contains available version numbers, history of package-changes and revision, distribution and urgency of the package
- File *install* → Initial definition of destination paths on the target computer. Copies every single file and folder to its correct destination.
- *\$paket.files*, *\$paket.dirs*, *\$paket.install* → connects the copied files and folders to the Debian-packages. This technique allows updates of single packages and therefore single files and folders
- *\$paketname.preinst*, *\$paketname.postinst* → definition of tasks and dependencies which are performed before and after installation
- *dpkg -build* → Tool to build the Debian-packages from the Maven-built BIRO software

When the configuration of the Debian-package is finished, then the actual build process can be triggered issuing the command:

```
$ dpkg -buildpackage
```

7.1.2 Server-Side: Creating a Linux Software Repository

Software repositories are used to manage and store software packages including their metadata to be installed on Linux systems, provided that the client has properly configured additional software repositories.

Repositories shall provide the possibility to manage software packages created for multiple distributions, architectures and software-package versions. Many different tools to create Debian-repositories are available³⁰. Amongst these, *reprepro*³¹ is a useful and easy to use tool to create APT-conform repositories.

Reprepro makes use of the file system for local use of the software-packages or a webserver, where the software-packages are stored, to distribute the software

²⁹ APT How-to. 19th of August 2010, from <http://www.debian.org/doc/manuals/apt-howto/index.en.html>

³⁰ How to setup a Debian repository. 19th of August 2010, from <http://wiki.debian.org/HowToSetupADebianRepository>

³¹ Reprepro – tool to handle local repositories. 29th of July 2010, from mirrorer.alioth.debian.org

publicly. A repository can be created anywhere, obviously a subfolder of a webserver is recommended to share the repository, e.g. `/srv/www/htdocs/apt`. A subfolder called *conf* is needed where the configuration is located.

This configuration file, called *distributions*, is needed to specify which releases the repository will contain. Configuration parameters are:

- Origin and label of the Software
- Codename, i.e. the distributions provided in the repo
 - stable (currently Lenny), testing (currently squeeze), unstable (sid) in **Debian**
 - currently: lucid, lucid-security, lucid-updates, lucid-packports in **Ubuntu**
- version numbers of the software package
- architectures
 - e.g. x86, AMD64, all, source
- components
 - main, non-free, contrib. in **Debian**
 - main, multiverse, restricted, universe in **Ubuntu**
- Description of the software package

Including a Debian-Package into a repository can be simply done by importing the *.deb*-Packages or the *.changes* file, produced by the building-tool, into the repository using the `reprepro` tool:

```
$ reprepro -b . include sarge filename
```

7.1.3 Client-Side: Getting Debian-Packages

Linux uses several techniques for (re-)installing, upgrading and removing software packages to an installation. This collection of tools is called package management system. A use case for installing new software is shown in Figure 12.

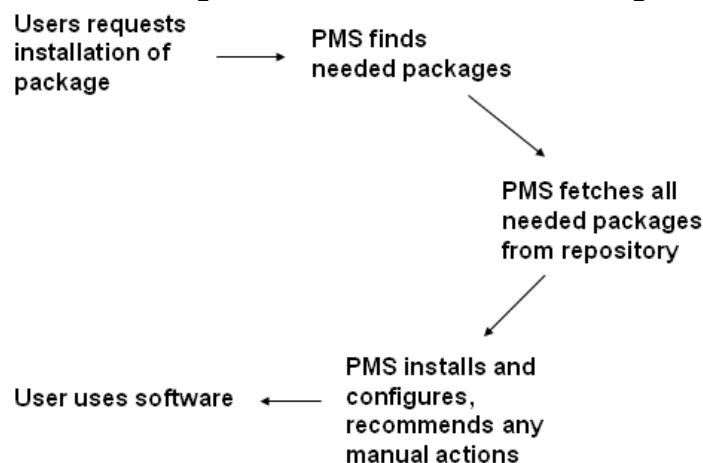


Figure 12 – Use Case for installing new software packages

Linux-packages exist in different formats, e.g. *deb*, *RPM* or *targ.gz/tgz*.

A very widely used format for packages is the Debian-package format (*deb*), which can be used on many different Linux distributions based on Debian and its derivate

Ubuntu. Non-Debian-based packages, like rpm used in Red Hat Linux distribution or Slackware's tgz, can be converted to Debian-packages using `alien`³².

In order to find software-packages stored in Debian software-repositories, which are not initially listed in the repository configuration-file, the file `/etc/apt/sources.list` (Debian-based) or any similar configuration file has to be modified properly:

```
deb http://www.example.com/apt sarge main contrib non-free
deb-src http://www.example.com/apt sarge main contrib non-free
```

Listing 1 – Example of an entry in `/etc/apt/sources` for an additional software repository

Remote located Debian-packages can be easily installed via software like `apt` (advanced packaging tool) and `aptitude`.

An alternative to the tools mentioned before can be the manual installation of Debian-packages. They can be installed without touching the software repository configuration. For this purpose the packages have to be downloaded manually and installed using the

```
$ dpkg -i Debian.package
```

command in the console or `gdebi` as graphical Debian-package viewing program.

7.2 Results

7.2.1 BIRO Debian-Packages

The concept of Debian-Packages containing the software of the BIRO system, allows users to use the BIRO system without necessarily using BIROX. The BIRO software uses Apache Maven³³ as project-, dependency-, build- and release-management tool. Using Apache Maven, build processes can be modified to meet the architectural requirements of Debian-packages.

The build-process consists of different sub-builds according to the multi-module project structure using Super POM inheritance, as mentioned in Figure 13, for the different pieces of the BIRO software. Figure 13 shows the build structure as created with Maven.



Figure 13 – Build structure of the BIRO System

³² Alien. 19th of August 2010, <http://kitenet.net/~joey/code/alien/>

³³ Apache Maven. 17th July 2010, from <http://maven.apache.org/>

BIROBox-2.0.7 is the library managing the invocation of the BIRO software. It uses third party Java-libraries as well as Java-libraries created within the subprojects *BIROAdaptor*, *BIROCommunicationSoftware* and *BIRODatabaseManager*. Every single library used is located in the *BIROCommonLibrariesFolder*.

The folder *_cs_* contains additional resources for the *BIROCommunicationSoftware* whereas the folders *lib* and *_se_* provide resources for the *BIROStatisticalEngine*.

The BIROBox can be started in the command-line using the command:

```
$ java -jar BIROBox-2.0.7.jar
```

For the creation of the Debian-packages some configuration parameters, as mentioned in chapter 7.1.1 - Building Software Packages for Debian, are necessary for the build tool *dpkg*. An example of the necessary configuration file for the Debian-packages for BIRO is explained below.

The *control* file describes which packages shall be built and which dependencies the packages will have among each other:

```
Source: biro
Section: java
Priority: extra
Maintainer: Matthias Wieser <matthias.wieser@joanneum.at>
Build-Depends: debhelper (>= 7)
Standards-Version: 3.8.1
Homepage: http://www.birobox.org

Package: birobox
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, libbiro-common-java, biro-communicationsoftware, biro-databasemanager, biro-statisticalengine, biro-adaptor2
Description: Birobox software
    Birobox software including all important libs and dependencies.

Package: libbiro-common-java
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, sun-java6-jre, birobox
Recommends: birobox
Description: Birobox software
    Common Libraries of the Birobox software.

Package: biro-communicationsoftware
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, birobox
Description: Birobox software
    BIRO Communication software including all important libs.

Package: biro-databasemanager
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, postgresql-server, birobox
Description: Birobox software
    Biro Database manager including all important libs.

Package: biro-statisticalengine
Architecture: any
Depends: ${shlibs:Depends}, ${misc:Depends}, birobox, r-cran-sp, r-cran-epi, r-cran-rjava, r-cran-lattice, r-cran-dbi, r-cran-hmisc, r-cran-maptools, texlive-tex-base, libcairo2-dev, libxt-dev
Description: Birobox software
    Biro Statistical Engine including all important libs and dependencies

Package: biro-adaptor2
Architecture: any
```



```
Depends: ${shlibs:Depends}, ${misc:Depends}, birobbox
Description: Birobbox software
Biro Adaptor including all important libs.
```

According to the package name, e.g. *birobbox*, the build-tool looks for files called *birobbox.dirs*, *birobbox.files* and *birobbox.installs*. An example of *birobbox.files* containing files needed for the package and their corresponding location on the destination system is given below:

```
usr/bin
usr/bin/birobbox-gui
usr/share/birottools/BIROBox-2.0.7.jar
usr/share/birottools/images
usr/share/birottools/images/biro_logo2.jpg
usr/share/birottools/images/biro_logo.jpg
```

After invocation of the Debian-package tool "*dpkg -i build*", the packages

- *biro-adaptor2_i386.deb*
- *biro-communicationsoftware_i386.deb*
- *biro-databasemanager_i386.deb*
- *biro-statisticalengine_i386.deb*
- *birobbox_i386.deb*
- *libbiro-common-java_i386.deb*

are created.

To make the Debian-packages available for installation on client's Linux systems, it is necessary to import the packages or the changelog file in the *reprepro* repository according to 7.1.2 - Server-Side: Creating a Linux Software Repository.

7.2.2 R Debian-Packages

The BIRO process includes the creation of statistical objects based on statistical calculations, i.e. tables, charts, graphs and figures. For this reason it's necessary to install the BIRO system with a dependency on the software R³⁴ which is a freely available software environment for statistical computing and graphics. Its widespread usage results in very tight release cycles - since April 2008 (version 2.7.0) until October 2010 (version 2.12.0) R has experienced several major upgrades. The development of the statistical software in the BIRO project was conducted using R version 2.8.0. Since a major modification in the R functionality in the upgrade to version 2.9.0 resulted in an incompatibility with the BIROStatisticalEngine, the source code can not be executed in a higher version than 2.8.0 - for the time being. Unfortunately there exists a discrepancy between this R version needed and the R version available in the different package management systems for the most recent distributions of Linux derivatives.

To resolve this dependency of the BIRO Debian-Packages, the required Debian-Packages of R for Ubuntu were downloaded from the "*Comprehensive R Archive Network*" (CRAN)³⁵ and repackaged in Debian-Packages each starting with the prefix "*biro-*".

Two positive effects arise from this technique:

³⁴ The R Project for Statistical Computing. 2nd of November 2010, from www.r-project.org

³⁵ The Comprehensive R Archive Network. 2nd of November 2010, from cran.at.r-project.org

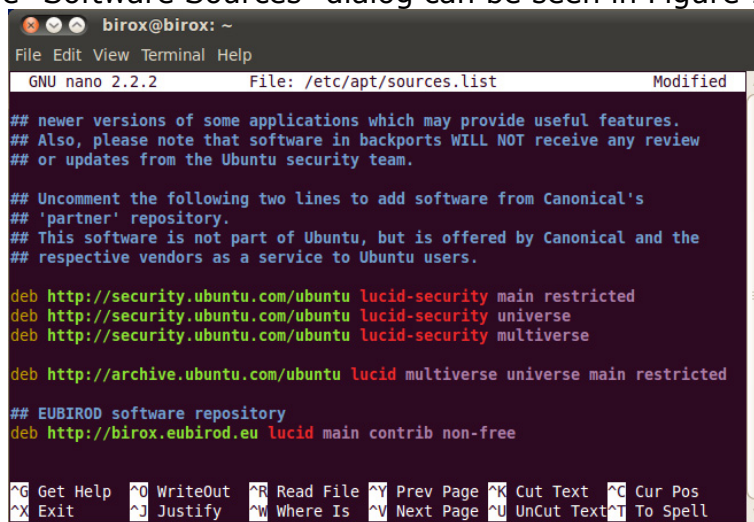
- The packages can be hosted on a software repository without interfering with existing R-packages from official Ubuntu repositories.
- Different versions of R can coexist on Linux due to the fact that R stores versions of the base-environment and used versions of the R packages

7.2.3 Using the BIRO Software repository

In order to install the BIRO system on any Linux it is necessary to add the software repository which manages the Debian-packages of the BIRO software-packages. The BIROX in its latest revision is already preconfigured with the software source, so this step only affects existing Linux systems, where the BIRO system shall be used.

The necessary steps to add additional software sources are explained assuming the existing Linux distribution is Ubuntu. For different distributions the manual has to be checked whether Debian-packages can be installed.

Additional software sources can be either added manually to the file */etc/apt/sources.list* using a text editor (see Figure 14) or using the Software Sources dialog found under **"System → Administration → Software Sources"**. A screenshot of the "Software Sources" dialog can be seen in Figure 15.



```

birox@birox: ~
File Edit View Terminal Help
GNU nano 2.2.2 File: /etc/apt/sources.list Modified

## newer versions of some applications which may provide useful features.
## Also, please note that software in backports WILL NOT receive any review
## or updates from the Ubuntu security team.

## Uncomment the following two lines to add software from Canonical's
## 'partner' repository.
## This software is not part of Ubuntu, but is offered by Canonical and the
## respective vendors as a service to Ubuntu users.

deb http://security.ubuntu.com/ubuntu lucid-security main restricted
deb http://security.ubuntu.com/ubuntu lucid-security universe
deb http://security.ubuntu.com/ubuntu lucid-security multiverse

deb http://archive.ubuntu.com/ubuntu lucid multiverse universe main restricted

## EUBIROD software repository
deb http://birox.eubiroad.eu lucid main contrib non-free

^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

```

Figure 14 – Configuration-file for Aptitude to manage the software repositories

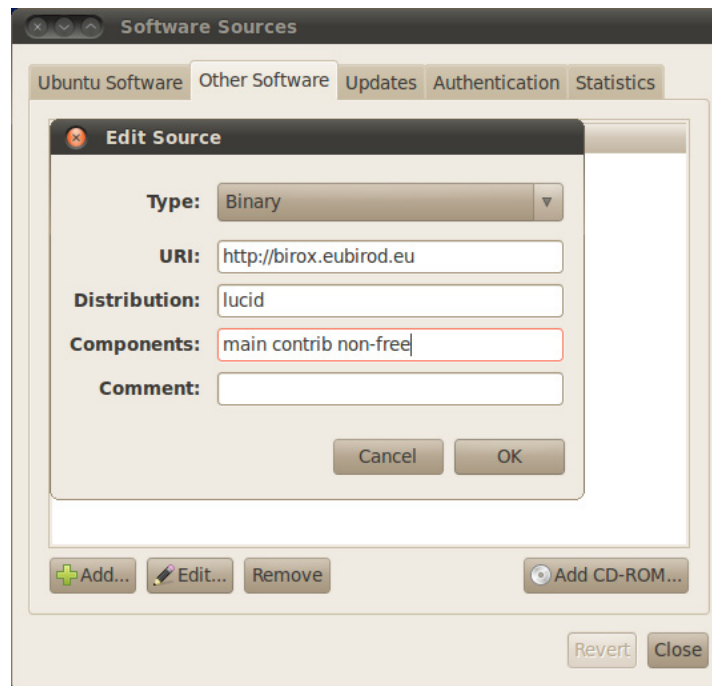


Figure 15 – Configuration wizard for Aptitude to manage software repositories

An update of the software sources is necessary, using the command:

```
$ sudo aptitude update
```

Once the software sources are setup and up to date, the BIRO system can be searched or installed using *aptitude*. The repository can be searched for the term "biro" using the command "*aptitude search biro*". Figure 16 shows the Debian-packages found with this term, i.e. Birobox-related packages and R-related packages.

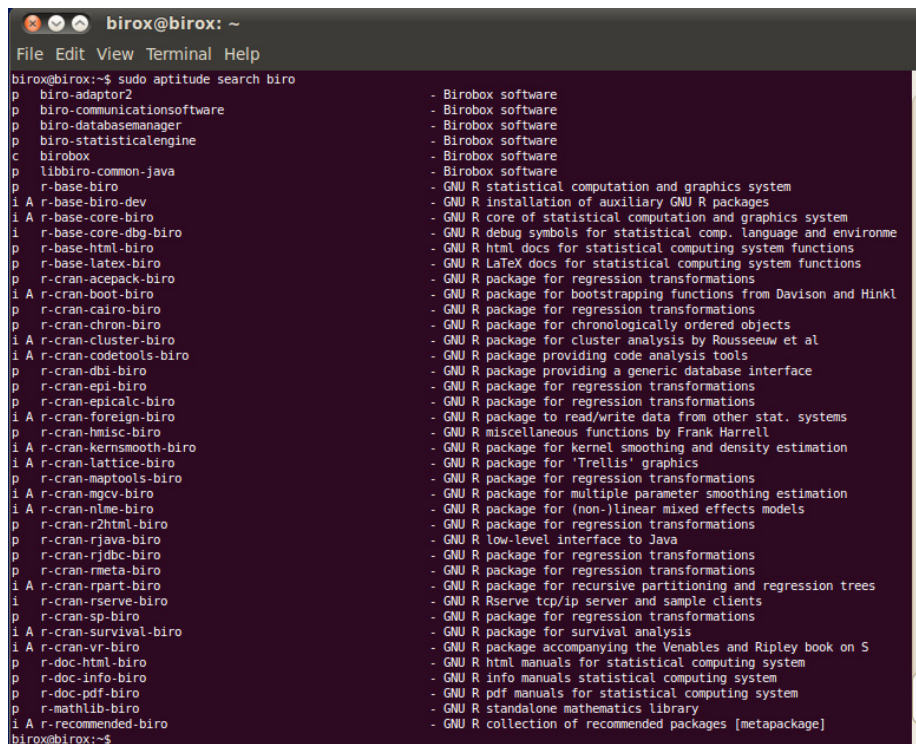
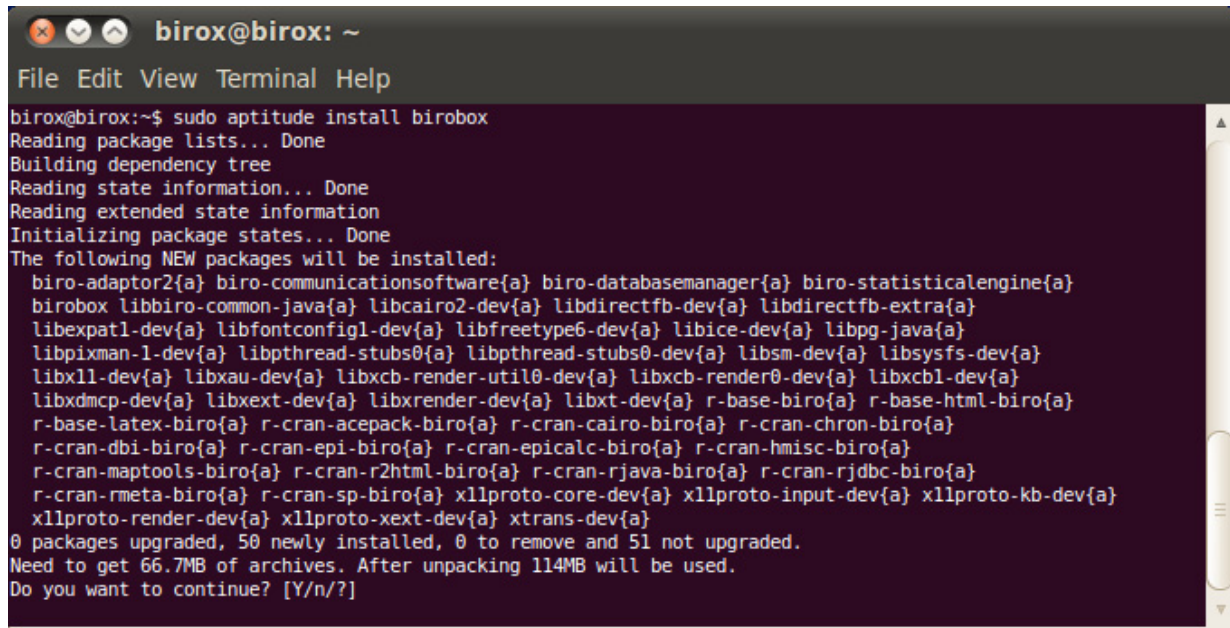


Figure 16 – Using aptitude to search the software repositories for the term "biro"

The subcommand *install* of *aptitude* installs the whole BIRO system including its dependencies on *Postgres*, *R*, *Latex*, *Java* and (optionally) *Pentaho*:

```
$ sudo aptitude install birobox
```

The BIRO system will be downloaded from the remote repository and installed on the client's system. See Figure 17.



```

birobox@birobox: ~
File Edit View Terminal Help
birobox@birobox:~$ sudo aptitude install birobox
Reading package lists... Done
Building dependency tree
Reading state information... Done
Reading extended state information
Initializing package states... Done
The following NEW packages will be installed:
  biro-adaptor2{a} biro-communicationsoftware{a} biro-databasemanager{a} biro-statisticalengine{a}
  birobox libbiro-common-java{a} libcairo2-dev{a} libdirectfb-dev{a} libdirectfb-extra{a}
  libexpat1-dev{a} libfontconfig1-dev{a} libfreetype6-dev{a} libice-dev{a} libpg-java{a}
  libpixman-1-dev{a} libpthread-stubs0{a} libpthread-stubs0-dev{a} libsm-dev{a} libsysfs-dev{a}
  libx11-dev{a} libxau-dev{a} libxcb-render-util0-dev{a} libxcb-render0-dev{a} libxcb1-dev{a}
  libxdmcp-dev{a} libxext-dev{a} libxrender-dev{a} libxt-dev{a} r-base-biro{a} r-base-html-biro{a}
  r-base-latex-biro{a} r-cran-acepack-biro{a} r-cran-cairo-biro{a} r-cran-chron-biro{a}
  r-cran-dbi-biro{a} r-cran-epi-biro{a} r-cran-epicalc-biro{a} r-cran-hmisc-biro{a}
  r-cran-maptools-biro{a} r-cran-r2html-biro{a} r-cran-rjava-biro{a} r-cran-rjdbc-biro{a}
  r-cran-rmeta-biro{a} r-cran-sp-biro{a} x11proto-core-dev{a} x11proto-input-dev{a} x11proto-kb-dev{a}
  x11proto-render-dev{a} x11proto-xext-dev{a} xtrans-dev{a}
0 packages upgraded, 50 newly installed, 0 to remove and 51 not upgraded.
Need to get 66.7MB of archives. After unpacking 114MB will be used.
Do you want to continue? [Y/n/?]

```

Figure 17 – Using aptitude to install the BIRO software-packages

The distribution of the complete BIRO software using Debian-packages has the advantage that every change made in the software by the development team and its ensuing build with maven and Debian-package manager (dpkg) results in a direct update-notification in the Linux where the software is installed. The user will be visually notified in the notification area of the top bar in Gnome³⁶. A screenshot of a pending software update can be seen in Figure 18.

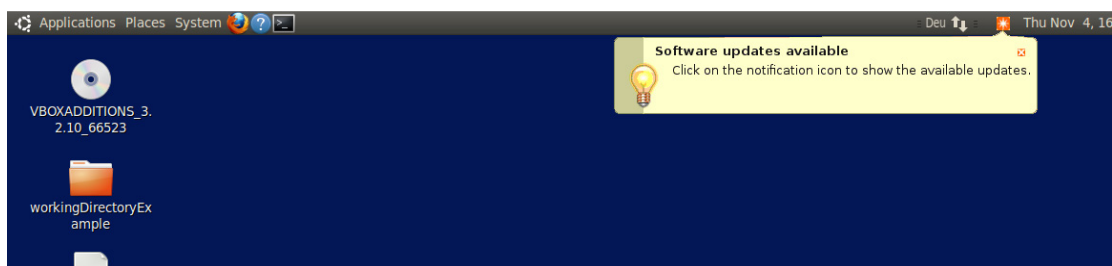


Figure 18 – Update notification of installed software in Gnome Desktop

The software can be easily updated when clicking on the notification area or using the terminal with the command:

```
$ sudo aptitude upgrade
```

³⁶ Gnome: The Free Software Desktop Project. 25th of August 2010, from www.gnome.org

7.2.4 Out-of-the-box solution: BIROX

In addition to the steps and techniques described in the previous chapters, which only require an existing *Ubuntu Lucid* installation, a preconfigured Linux distribution (*BIROX*) was created which supports the whole BIRO process to provide an even more out-of-the box solution for non-technical skilled people.

As base for BIROX, two different approaches were tested to create this solution:

In the first test, a slim derivate of a **Debian**-Linux was used as base to setup the BIRO system. The BIRO system was installed on using a custom shell script which copied the files of the BIRO software to the operating system where they were needed. Since BIROX is created mainly for users like physicians or other health professionals the focus of the operating system was set on usability and software-/hardware-support. Debian did not meet these requirements so another solution was tested.

In the second test, a full featured distribution of **Ubuntu**'s newest release *Lucid Lynx* was used to setup the BIRO environment. Ubuntu comes with neat features concerning the user interface and soft- and hardware-integration. Furthermore, Ubuntu's Debian-based PMS *apt-get* and *aptitude* provide full featured support with the technique described in chapter 7.1.1.

A screenshot of the first release of BIROX based on **Ubuntu** can be seen in Figure 19.



Figure 19 - DVD label of the BIROX distribution

Using Live CD Customization, it is possible to create the Ubuntu distribution according to someone's needs, where software packages can be added, removed or updated as well as system defaults like themes, icons, desktop backgrounds and localisations can be changed.

A complete reference, with the steps needed for the customization, is given here³⁷. Especially some particular tools for the BIRO system have been installed during customization:

³⁷ LiveCDCustomization. 31st of October 2010, from <https://help.ubuntu.com/community/LiveCDCustomization>

- Pentaho Data Integration Spoon 4.0.1
- Adobe Reader 9
- Sun Java 6
- Mozilla Firefox Webbrowser

For the operation of the BIRO system it is necessary to build and start the customized Ubuntu image for the first time using *qemu* to make some modifications to the configuration of the Linux itself and to some software packages, i.e.:

- Add the BIRO software repository to the *sources.list* according to chapter 7.2.3.
- Install BIROBox from the <http://birox.eubirod.eu> repository. This includes the automatic installation of its dependencies, i.e. R, Latex, Postgres, PgAdmin III.
- Copy an example of a valid working directory containing a sample configuration and a sample dataset to the Desktop of the *birox* user, so one can start the BIRO process immediately
- Create a *birox* user and a *birox* database on the installed Postgres-database (user: **birox**, password: **birox**)
- Set EUBIROD link for Mozilla Firefox on the Desktop for a quick lookup of references

BIROX is distributed via the project's homepage³⁸ as an ISO-image, which can be burned on DVD to be run as Live-DVD or as Setup-DVD for permanent installation.

When the **Live-DVD** is used, the recommended way to provide data to the system is a **USB flash drive** or memory stick. It is not recommended or, for some file-systems, not possible to access the stationary hard disks to store data permanently.

When **permanent setup** is used, the operating system has full control over the host PC, so data can be copied to a **stationary hard disk**, processed there and stored permanently.

8.Executing BIROX on a Virtual PC

Another possibility to run BIROX is with the use of virtualisation. Subsequently this technique is described.

8.1 Background

Software installations, as mentioned in the chapters before, interfere with the original operating system they are installed on. Not in every case, as it was observed during the software tests on the operating systems of the participating partners, the installation and operation could successfully be performed. A technology called virtualization³⁹ encapsulates the BIRO system from the host

³⁸ The EUBIROD project website. 24th of August 2010, from www.eubirod.eu

³⁹ Virtualization in this context is limited to System Virtualization, in contrast to Process Virtualization. For a detailed description please visit the article: Virtual Machine. In *Wikipedia*. Retrieved October 25th 2010, from http://en.wikipedia.org/wiki/Virtual_machine

system and provides a possibility to run the BIRO system preinstalled on an autonomous operating system within an existing operating system. Virtualization allows the sharing of physical machine resources between different virtual machines, each running its own operating system.

Virtualization is generally useful for several scenarios:

- **Operating system support.** With Virtualization, one can run software written for one operating system on another (for example, Windows software on Linux or a Mac) without having to reboot to use it.
- **Testing and disaster recovery.** Once installed, a virtual machine and its virtual hard disks can be considered a “container” that can be arbitrarily frozen, woken up, copied, backed up, and transported between hosts. At every point in time a snapshot of the current state of a virtual machine can be taken, so it’s possible to switch between states of virtual machines.
- **Infrastructure consolidation.** Virtualization can significantly reduce hardware and electricity costs. So, instead of running many such physical computers that are only partially used, one can pack many virtual machines onto a few powerful hosts and balance the loads between them.
- **Easier software installations.** Virtual machines can be used by software vendors to ship entire software configurations. With virtualization it becomes possible to ship an entire software solution, possibly consisting of many different components, in a virtual machine, which is then often called an “appliance”.

Several virtualisation techniques exist, namely VirtualBox⁴⁰ and VMware⁴¹, however VirtualBox was used for the BIRO System to create a Virtual Machine.

8.2 Results

A virtual PC alternative of BIROX is distributed via the project’s homepage⁴² as a Virtual Disc Image (VDI), which can be run in a virtual environment like VirtualBox.

A complete reference, how to setup up a virtual environment in Virtual Box, is given in the section Usage of BIROX in Oracle VirtualBox in Appendix A.

The recommended way when using a **Virtual Disc Image**, is to make data available to the virtual PC using an **exchange directory** as described thoroughly in Appendix A - Configuration of the Exchange Directory.

A second option would be using USB memory sticks or flash drives plugged into the computer on the virtual guest system, but unfortunately this didn’t work in the current version of Virtual Box. Since this option already worked in former versions either an incompatibility with the used Ubuntu version or a bug in Virtual Box could be the reason for this problem.

⁴⁰ VirtualBox. 25th of October 2010, from www.virtualbox.org

⁴¹ VMWare Virtualization Software. 25th of October 2010, from www.vmware.com

⁴² The EUBIROD project website. 24th of August 2010, from www.eubirod.eu

9. Extended Data Transformation

9.1 Materials and Methods

The BIROBox provides a container to run all processes of the BIRO system within one tool. One of these steps is to connect to a local data source, to extract data and to transform data in the format of the BIRO Common-Dataset^x and its revised successor^{xi} of the EUBIROD project. Subsequently, the BIRODatabaseManager will load the data in the BIRO database so the BIROStatisticalEngine can run statistical analyses on the data and create reports. Some limitations of the currently implemented solution were revealed which will be described in the following chapter.

9.1.1 Background and Limitations

When the BIROAdaptor is connected to a data source (databases or csv-Files), the BIROBox displays a GUI-dialog, where users can map local fields to each corresponding BIRO field. This mapping can also be combined with simple transformation-steps. A screenshot of this dialog can be seen in Figure 20.

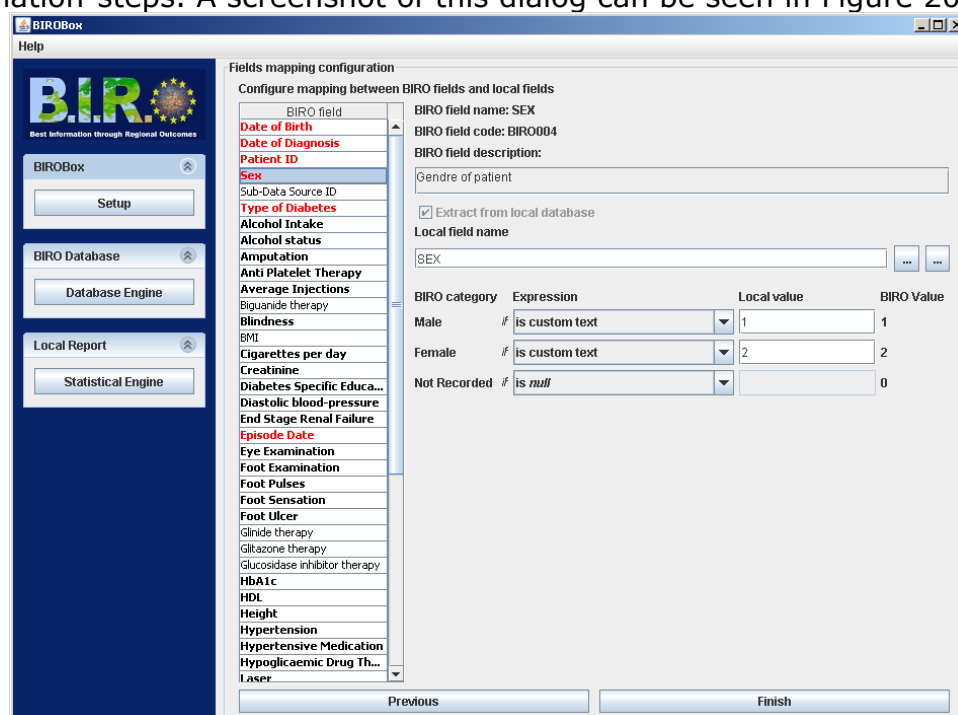


Figure 20 – Mapping between data source fields and BIRO data items within the BIROBox. The BIROAdaptor uses the information of the mapping for the transformation to the format of the BIRO Common-Dataset.

This solution of data-mapping fits best if the structure of the data does not differ vastly from the structure of the **Merge-Table**, which is a database structural representation of the BIRO Common Dataset. Regarding the quality of data provided from national data-sources and the data diversity of these sources (see Appendix D) the challenges for the BIROAdaptor concerning data integration were bigger as initially assumed. A proper transformation of data with these characteristics is unfortunately not possible in every case using the routines implemented in the BIROBox so far.

Another important aspect is the fact that not every dataset can be provided in the same granularity⁴³ as the BIRO Common-Dataset. The possibilities of the transformation routines, implemented in the adaptor, are not sufficient to split or combine values. A possibility could be, when a BIRO data field is computed out of other local fields, like:

- **BMI** uses height and weight to be computed
- **Retinopathy-/ Foot-Ulcer status** is calculated using data fields for either the left or the right side of the human body
- **Smoking status** can derive of cigarette intake: e.g. more than a certain threshold [cigarettes/day] means patient is a smoker
- **Alcohol status** can derive of alcohol intake: e.g. more than a certain threshold [g of alcohol/day] means that the patient is a currently a drinker
- **Lipid Lowering Therapy** can derive of the information if a patient is treated medically against dyslipidemia and whether the cholesterol value is increased over a certain threshold.

These requirements for transforming source data into the BIRO-format led to considerations to include alternatives and more flexible solutions.

One possibility can be to perform data transformations of the source data directly on the databases. For this purpose, it's recommended to create a database view which is able to alter the data-representation so it nearly matches the BIRO Common Dataset. An excerpt of a possible SQL-query including transformations using *CASE-WHEN*-expressions on the database side to create a **SQL-View** is given in Figure 21.

```
-- TYPE_DM (BIRO003)
CAST(dbo.TPatienten.sDiabetesTyp AS nvarchar(1)) AS TYPE_DM,

-- SEX (BIRO004)
CASE WHEN sGeschlecht = 'M' THEN 1 WHEN sGeschlecht = 'W' THEN 2 ELSE 0 END AS SEX,

-- DOB (BIRO005)
CAST(DATEPART(year, dbo.TPatienten.dGeburtsdatum) as nvarchar)+'-01-01' AS DOB,

-- DT_DIAG (BIRO006)
CASE WHEN nDiabetesbekanntseit19 IS NULL
THEN '1/1/1900' ELSE '1/1/' + CAST(nDiabetesbekanntseit19 AS varchar) END AS DT_DIAG,

-- EPI_DATE (BIRO007)
CAST(DATEPART(day, dbo.TSheets.dExpBegin) AS nvarchar) + '1/1/'
```

Figure 21 - Excerpt of a SQL-query including transformations to create a database view

A disadvantage using transformations on the database with views is the fact that not every partner in the project has direct access to the data on the server or the proper permissions to create a view. Moreover not every project partner is necessarily skilled enough to perform these tasks. These limitations should be minimized using another possibility to extend the BIROBox functionality.

In terms of mapping and transforming data, data integration techniques known from **Business Intelligence** (BI) can provide useful assistance. A key component

⁴³ Granularity, in this case, means that not every single BIRO field can be derived from an input field of the local dataset, e.g. no 1:1 mapping between data fields is possible.

of BI is to extract, transform and load (ETL) data from an external resource so data can be used in the desired format where it is needed. The BIROAdaptor, which provides mapping techniques as described above, does basically the same things without providing full functionality like third parties ETL-Tools, e.g. Pentaho Data Integration⁴⁴, Talend⁴⁵ or Informatica⁴⁶ PowerCenter.

ETL-Tools save time and money when it comes to hand-coded data integration logic. These tools are well-proven, validated and developed solutions can be reused in many different cases. The development team mainly focussed on Pentaho Data Integration for extended data transformation.

9.1.2 Pentaho Data Integration

Pentaho Data Integration (PDI), formerly known as Kettle, is a data warehousing component of the Pentaho Business Intelligence Suite. PDI is written in pure Java and consists of four components to provide ETL functionality:

- Spoon: Graphical data integration IDE⁴⁷ to design ETL processes, i.e. transformations and jobs. Many different data flow functions are available, like reading, validating, refining, transforming and writing data from and to different data sources
- Pan: an command-line tool dedicated to run data transformations designed in Spoon
- Kitchen: an application to execute jobs in a batch mode usually combined with a scheduler
- Carte: a web server which allows remote monitoring of running PDI ETL-processes through a web browser

A workflow of an ETL process consists of jobs, which call transformations, which itself define the dataflow using steps. A schematic overview of an ETL process is shown in Figure 22.

⁴⁴ Data Integration – ETL Tools & ETL Open Source Tools. 21st of August 2010, from www.pentaho.com/products/data_integration

⁴⁵ Talend: Open Source ETL and Data Integration Software. 21st of August 2010, from www.talend.com

⁴⁶ Informatica – Data Integration. 21st of August 2010, from www.informatica.com

⁴⁷ IDE...Integrated Development Environment

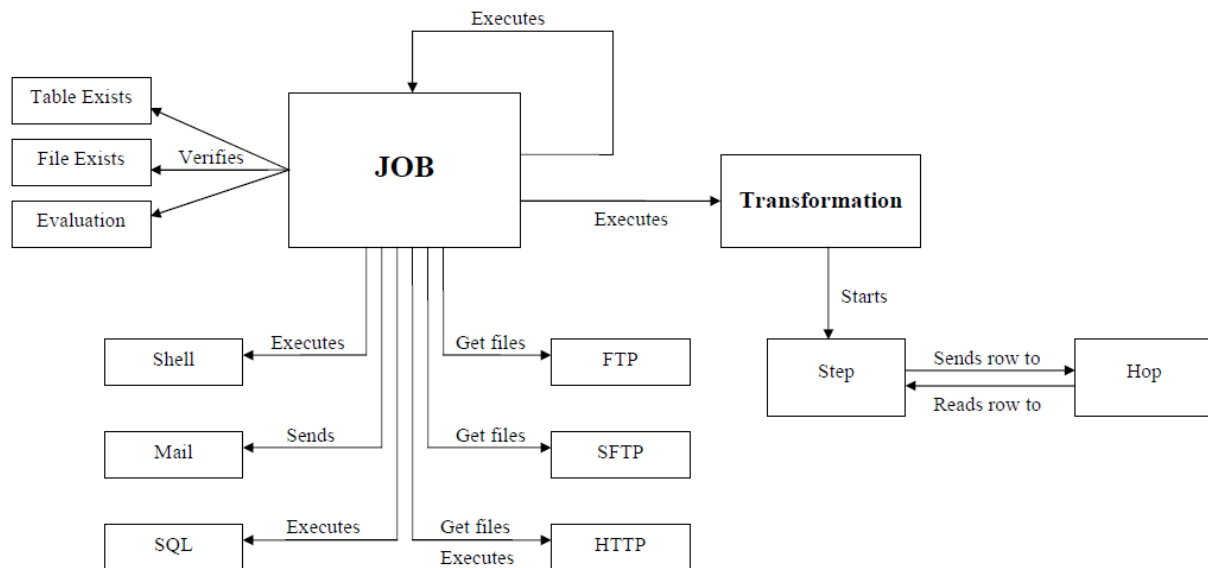


Figure 22 – ETL workflow in Kettle using jobs and transformations

Figure 23 shows an example of a screenshot of the graphical user interface Spoon, Pentaho's IDE for ETL-processes.

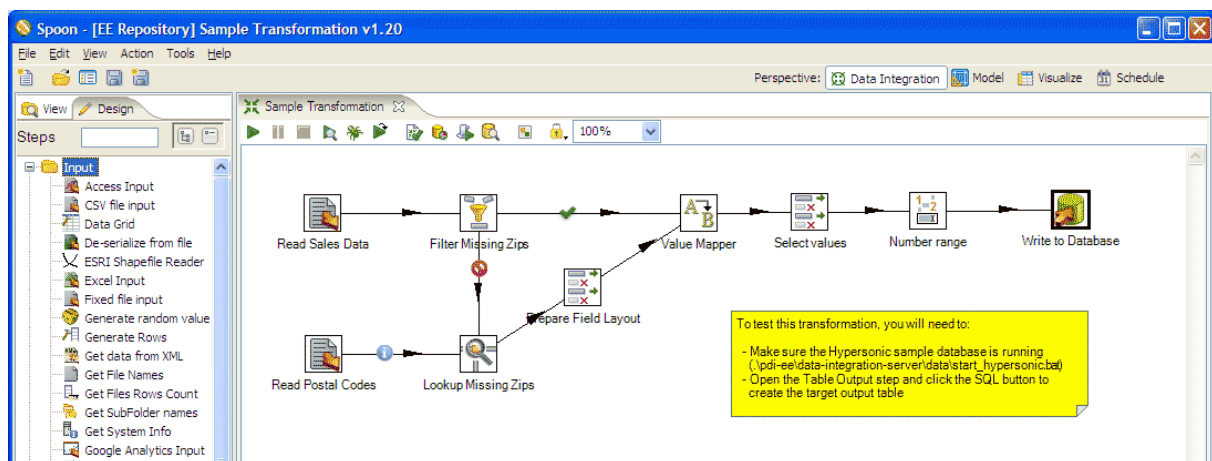


Figure 23 – example of a transformation in Kettle using the Pentaho Spoon workflow-designer⁴⁸

PDI provides the possibility to be executed as server application as well as stand-alone application. Furthermore it can be fully integrated into any Java project when the correct libraries of PDI are used. Transformations designed in Spoon are saved as xml-Files, which can be loaded by any tool of software which subsequently executes it via the PDI-API. This feature allows PDI and its corresponding libraries to be integrated into to BIROBox, so data integration techniques can be additionally performed by a third party tool, which is well-proven and validated.

⁴⁸ ETL workflow in Kettle using jobs and transformations. 23rd of August 2010, from http://www.pentaho.com/images/PDI_transformation.png

9.2 Results

As mentioned in the chapters above, Pentaho Data Integration can be used within existing Java software to run jobs or transformations. For this purpose PDI comes with the handy tool Spoon, which helps designing ETL-processes using a graphical user interface. Via drag and drop, process components can be added to a transformation.

For the BIRO software system a general PDI-process was designed which can be used within the BIROBox. The overall-process of this Pentaho job can be seen in Figure 24.

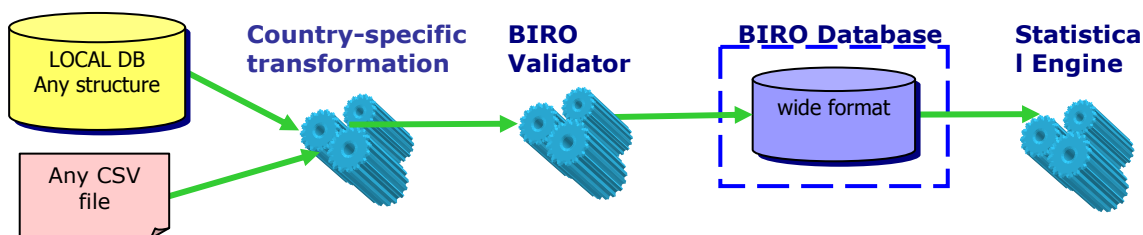


Figure 24 - PDI-job for the BIRO system

The job consists of a data-source, two transformations and a data-destination, which will be subsequently used by the statistical engine. The type of data-source depends on the infrastructure of the project-partner running the BIROBox. The partner has to design a country-specific transformation (Chapter 9.2.1) which connects to the data-source and maps the data to the correct format of the BIRO Common-Dataset. The second transformation, the BIRO-Validator (Chapter 5.4.2), checks the data-mapping according to the Common-Dataset making use of the validation step in Spoon and imports the data into the BIRO database in the correct format, i.e. wide format.

(Bouman and Dongen, 2009)^{xii} provides a complete reference of all features and a thorough description of Spoon and other Pentaho products. A complete reference of available transformation steps in Pentaho Spoon can be seen here⁴⁹

9.2.1 Country-specific transformation

Digging into to country-specific transformation, it is possible to design the transformation totally according to someone's needs. An example how a possible design of a transformation could look like is described in this chapter.

Figure 25 shows an example of a country specific transformation. The example shows a csv-File which is being loaded and altered using the "*Modified Java Script Value*"-technique. This step provides real fast features on a script-basis to modify values using Java-Script syntax. A screenshot of a dialog for designing this script can be seen in Figure 26. In this example the patient's height is converted from centimetres to meters and MA_TEST derives from the scalar value of micro albumin according to the Common-Dataset. Non-existing values are added as NULL-values to provide data-field consistency for the BIRO-Validator transformation.

⁴⁹ Pentaho Data Integration Steps - Pentaho Wiki. 30th of August 2010, from <http://wiki.pentaho.com/display/EAI/Pentaho+Data+Integration+Steps>

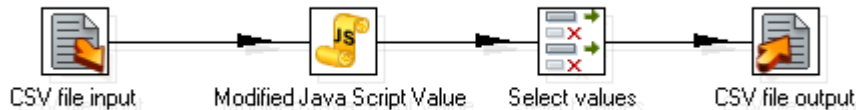


Figure 25 - Example of a country-specific transformation in PDI

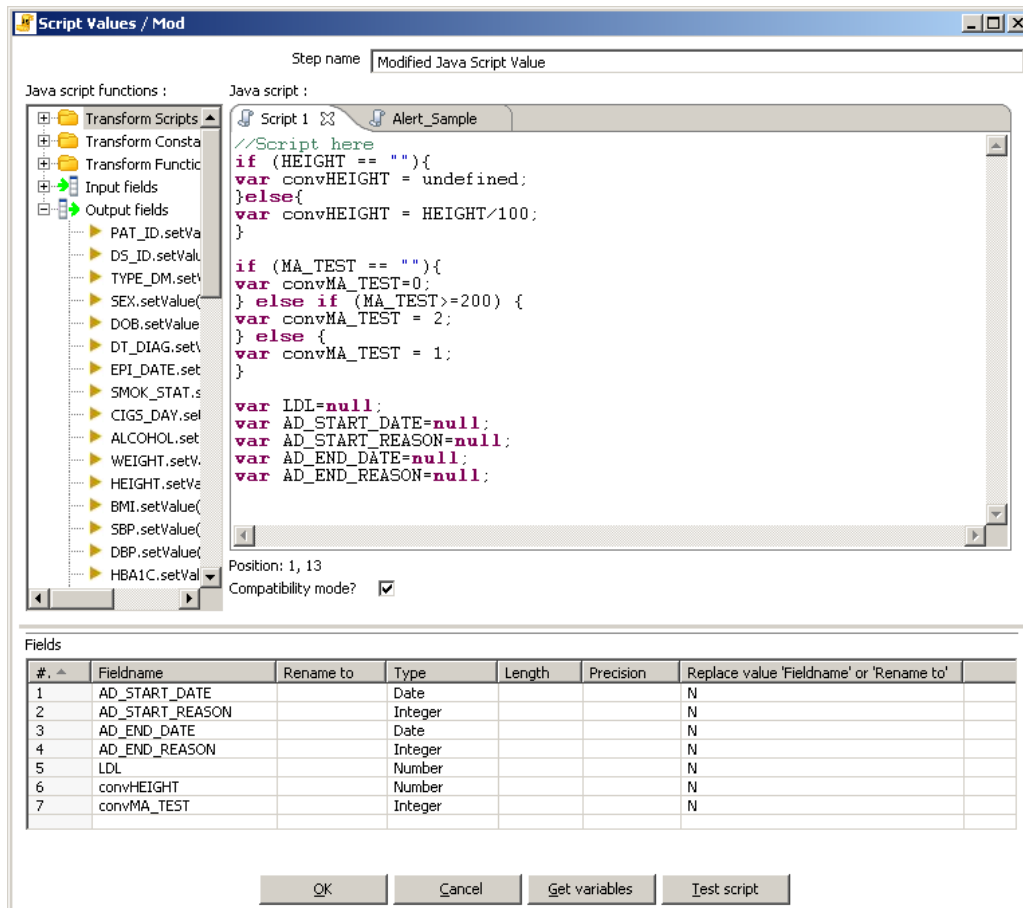


Figure 26 - "Modified Java Script Value"-dialog of PDI's Spoon to transform values of data fields really fast on a script basis.

The "Select Values"-step is used to select, rename, remove data-fields, to alter data types and to change the precision and length of data-fields. In the example the fields created in the "Modified Java Script Value"-step, i.e. *convHEIGHT* and *convMA_TEST*, are renamed to their supposed names according to the Common-Dataset. A screenshot of the "Select Values"-dialog can be seen in Figure 27.

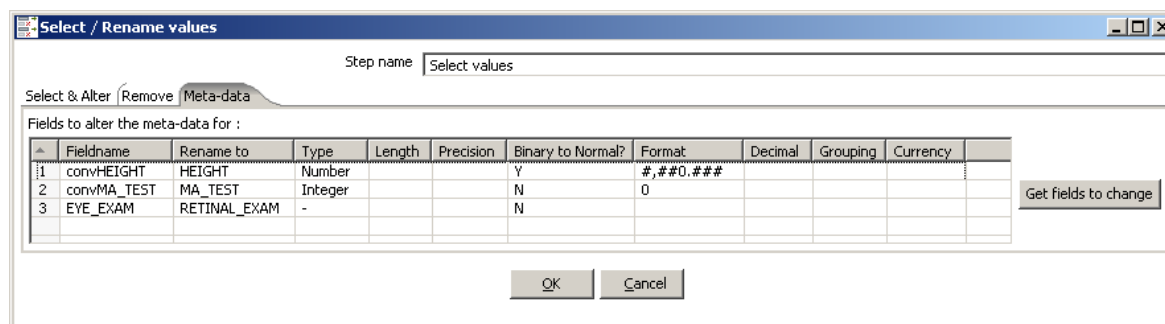


Figure 27 - Dialog which provides functionality to select, alter, remove or modify existing data fields

In the last step of the transformation, the resulting fields have to be written to an output stream, e.g. a file on the hard-drive, so the next transformation, i.e. BIRO-Validator, can use it as input for validation by automatically passing in the name of the generated csv-File via a variable.

9.2.2 BIRO-Validator

This part of the PDI-job is a transformation, provided by the EUBIROD software development team. This transformation loads the dataset, verifies the output of the country-specific transformation in terms of the Common-Dataset and writes the data-rows separated in valid and invalid data-fields. A screenshot of the whole process can be seen in Figure 28.

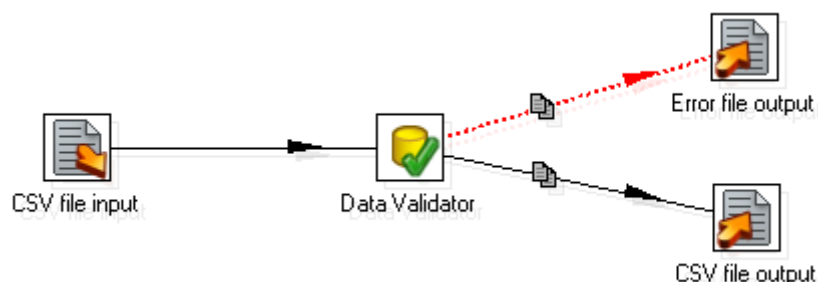


Figure 28 - BIRO-Validator transformation. The dataset is split into correct and incorrect data fields.

For this purpose the output, e.g. a csv-File will be loaded by PDI and the data-fields will be passed to the validation step called "Data Validator". This validation step is a direct mapping of the BIRO data items according the BIRO Common-Dataset^x and the EUBIROD revised Common-Dataset^{xi}. Every data-field will be checked on its name, data-type, possible constraints for scalar values, allowed values for enumerations or whether NULL values are allowed.

The definition of the BIRO data item "HEIGHT" is shown in Table 1. A screenshot of the corresponding data validation step showing the verification of the data-item definition in Pentaho Data Integration is shown in Figure 29.

| | |
|--------------------|--|
| Parameter: | Height |
| BIRO Ref: | BIRO012 |
| Field Name: | HEIGHT |
| Data Type: | Real (nnn.nn) |
| Definition: | Height in metres - measured without shoes. It is particularly important to |

| | |
|----------------------|--|
| | measure regularly the height of children. In adults a single recording will usually be sufficient. |
| Units: | M |
| Lower Range: | 0.4 |
| Upper Range: | 2.5 |
| Mandatory: | No |
| Validity: | High |
| Data Mapping: | Height measured in m = height in cm/100 |

Table 1 - Common-Dataset definition of the BIRO data item 012:'HEIGHT'

The screenshot shows the 'Data Validator' window with the following configuration:

- Stepname:** Data Validator
- Select a validation to edit:** BIRO012 (highlighted in the list on the left)
- Report all errors, not only the first:** ☒
- Output one row, concatenate errors with separator:** |
- Validation description:** BIRO012
- Name of field to validate:** HEIGHT
- Error code:** (empty)
- Error description:** (empty)
- Type:**
 - Verify data type?** ☒
 - Data type:** Number
 - Conversion mask:** (empty)
 - Decimal Symbol:** (empty)
 - Grouping Symbol:** (empty)
- Data:**
 - Null allowed?** ☒
 - Only null values allowed?** ☐
 - Only numeric data expected?** ☐
 - Max string length:** (empty)
 - Min string length:** (empty)
 - Maximum value:** 2,5
 - Minimum value:** 0,4
 - Expected start string:** (empty)
 - Expected end string:** (empty)
 - Not allowed start string:** (empty)
 - Not allowed end string:** (empty)
 - Regular expression expected to match:** (empty)
 - Regular expression not allowed to match:** (empty)
 - Allowed values:** (empty list with 'Add' and 'Remove' buttons)
 - Read allowed values from another step?** ☐
 - The step to read from:** (empty dropdown)
 - The field to read from:** (empty dropdown)

Buttons at the bottom: OK, New validation, Remove validation, Cancel.

Figure 29 - Data validation of the BIRO data item 012: HEIGHT

After the validation of every data-row, valid data-rows can either be saved as csv-File, like in the example, or saved directly in the database. At the current stage of PDI-integration in the BIROBox, the BIRODatabaseManager imports the dataset

into the BIRO database using the generated csv-File. In the next release of the BIROBox a direct data import into the BIRO database is supported, which makes the usage of the BIROAdaptor optional. The overall process of the data import within the BIROBox using PDI as well as the BIROAdaptor including data validation is illustrated in Figure 30.

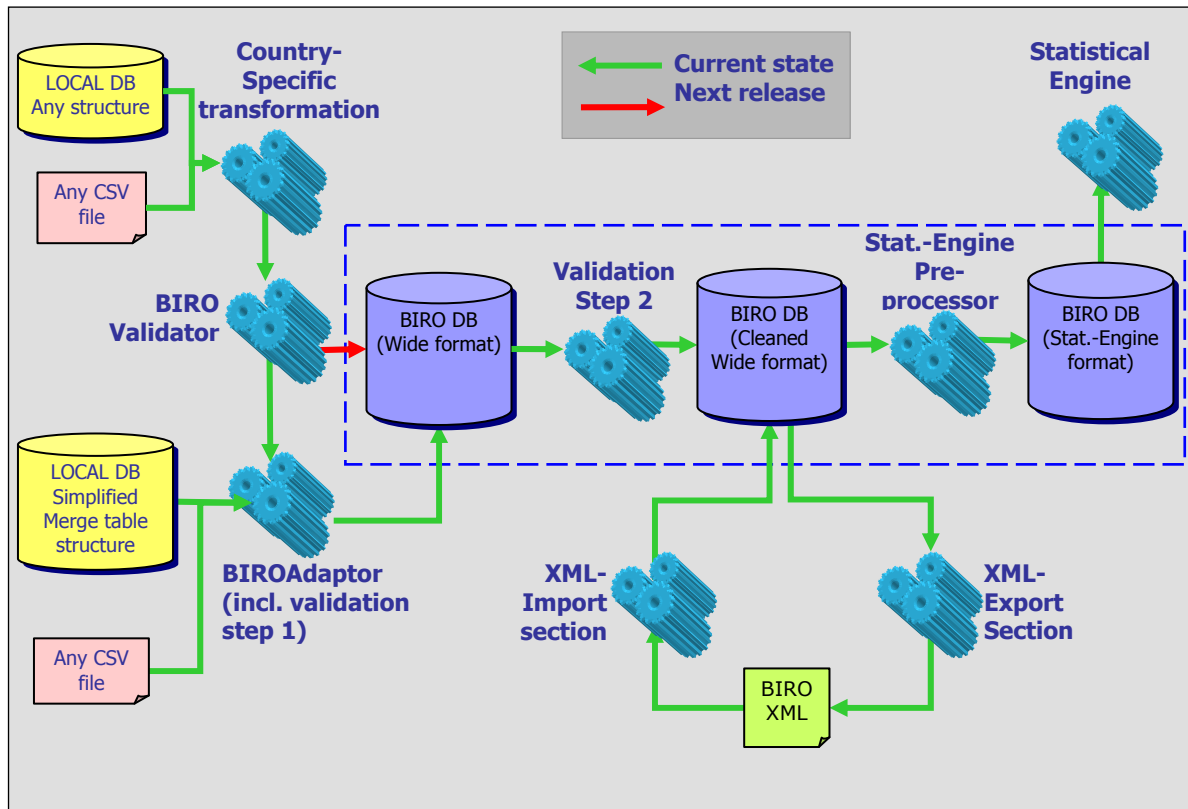


Figure 30 - Overall process of data import within the BIROBox

9.2.3 BIROBox Integration

ETL-transformations created in Pentaho Data Integration's Spoon have to be saved on the hard disc. They can be executed anytime using the PDI environment, i.e. Spoon, Pan, Kitchen or Carte or within the BIROBox when adding the correct dependencies to the Maven project.

For this purpose a dialog to trigger and correctly execute the PDI-transformations on the existing datasets is needed. The current release of the BIROBox provides a simple dialog to run jobs or transformations. This dialog is implemented in the Merge-Table configuration wizard. It contains a dialog option that allows users to select a csv-File, where the user can choose between the built-in transformation-routine using the BIROAdaptor or the customized configuration using the customized toolbox. A screenshot of this dialog can be seen in Figure 31.

For a more complex and more sophisticated usage of PDI an extended dialog to set different variables for PDI-configuration is needed.

Merge Table Source Configuration

☐ database

database name:

database table:

☒ **csv file**

csv file name *:

separator:

☐ xml file

archive file name:

Customized Toolbox

☐ Customized configuration available

Figure 31 - Merge-Table source configuration dialog using a csv-file and a customized configuration designed in PDI

The possibilities when using Pentaho Data Integration are nearly unlimited. Although PDI and especially Pentaho Spoon have some usability constraints, this data integration technique provides a fast, customizable, reusable and valid possibility to the data-integration feature already implemented within the BIROBox.

The runtime of data-transformation using PDI takes much less than the existing alternative, namely BIROAdaptor and BIRODatabaseManager. The validation logic implemented in the PDI BIRO-Validator is completely independent from the actual source code and built of the BIROBox and the BIROAdaptor. This results in a great advantage concerning the adoption of the data validation constraints, when the project consortium decides to make changes in the Diabetes Common-Dataset.

Moreover, if, as stated in the proposal of the project, the BIRO system should provide a generic framework for data acquisition for multiple chronic illnesses and therefore multiple datasets, it is absolutely beneficial to separate the common process of data import from the medical expertise of data-item constraints and its validation. Otherwise a complete adoption of the logic implemented in the software's source code and a subsequent rebuilt, release, update and further adoption of the software on the client side would be inevitable.

PDI and its IDE Pentaho Spoon can also be used in Linux. So it's possible for every user of the BIROX or any other Linux user, who installed the BIRO system using the Debian-packages, to design one's own transformation scenario for data integration to the BIRO system. Further activities run in the framework of WP7 will identify key issues in the management of data transformations and problems that can be easily solved through the development of customized Kettle transformations. Furthermore training in Pentaho Spoon for potential users can be useful during an upcoming EUBIROD meeting.

10. Appendix A

Download and Installation of Oracle VirtualBox

Use this link: <http://www.virtualbox.org/wiki/Downloads> to download Oracle Virtual Box and install a virtual BIROX. For now, the "Personal use and evaluation license (PUEL)" should be appropriate for us. – Otherwise you can also choose the Open Source version on the bottom of the page. It is equivalent but does not provide USB support.

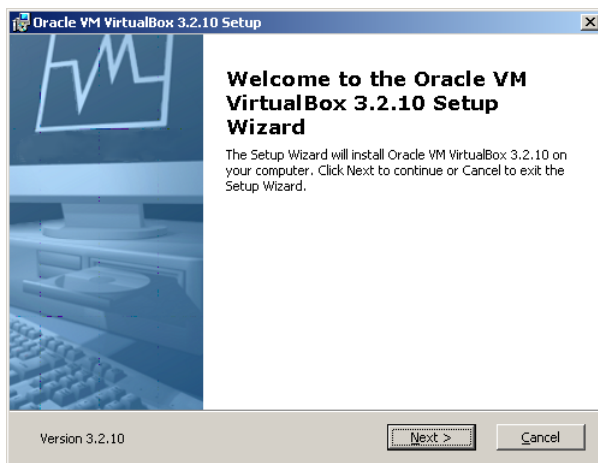


1. Choose the most current version for your operating system (Windows, Linux, ...)
2. Save the installation file "VirtualBox-3.2.10-66523-Win.exe" (or similar) on your computer

Execute the installation file with Administrator privileges.

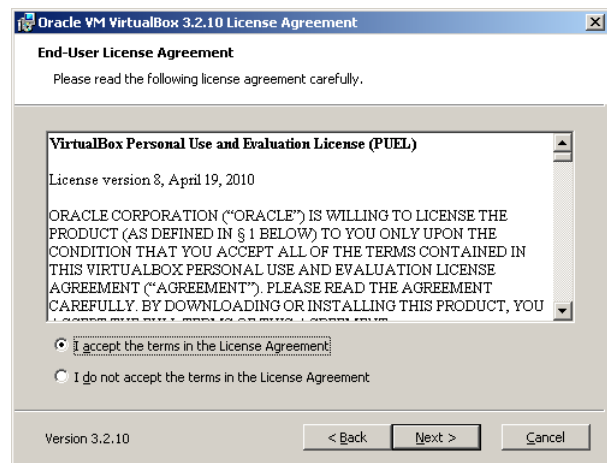
The following steps are explained for MS Windows OS⁵⁰.

1) Welcome



→ "Next"

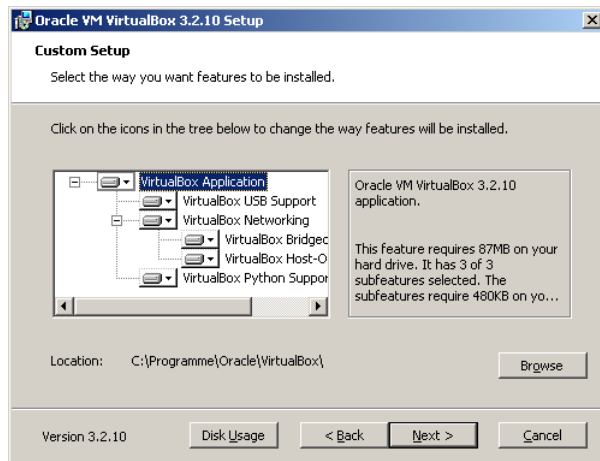
2) License Agreement



→ "Next"

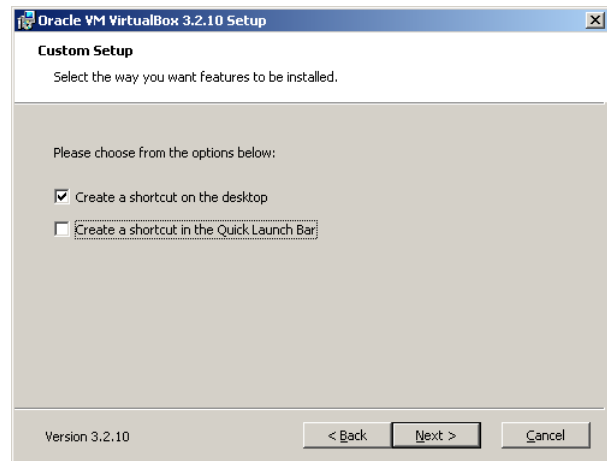
⁵⁰ If you use another OS, please proceed accordingly.

3) Custom Setup



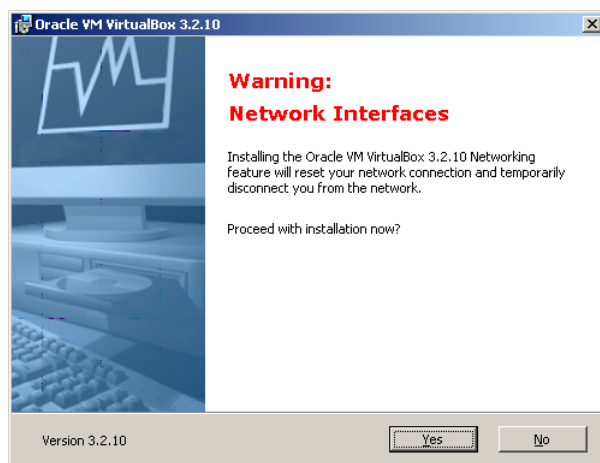
Install all packages (preset)
accept the location without any changes
→ **"Next"**

4) Shortcuts



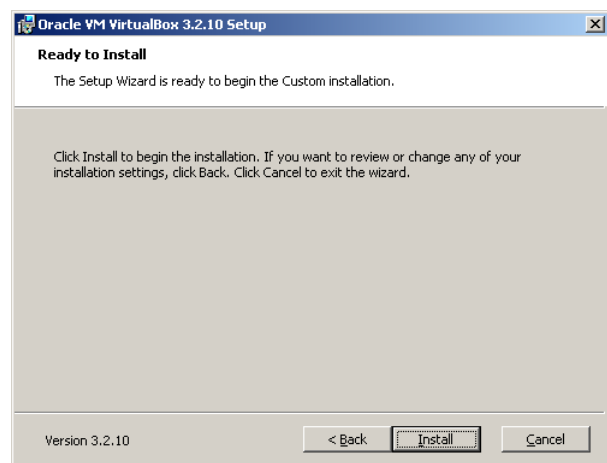
You can disable the Shortcut for the Quick Launch Bar.
→ **"Next"**

5) Warning (no changes required)



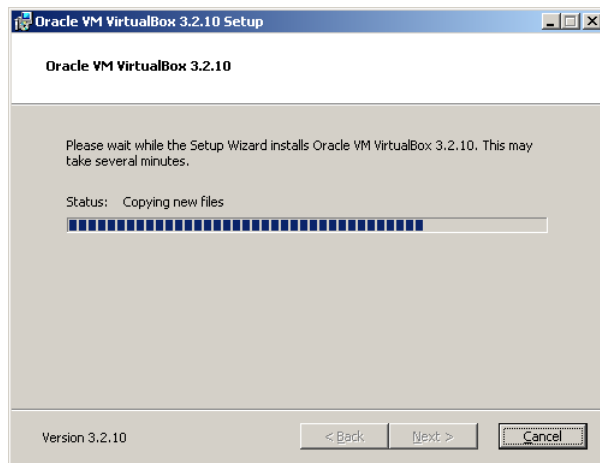
Better finish talks on Skype, downloads which may be running... before going on.
→ **"Next"**

6) Ready to install



Start installation by clicking on **"Install"**

7) Installation



(Wait)

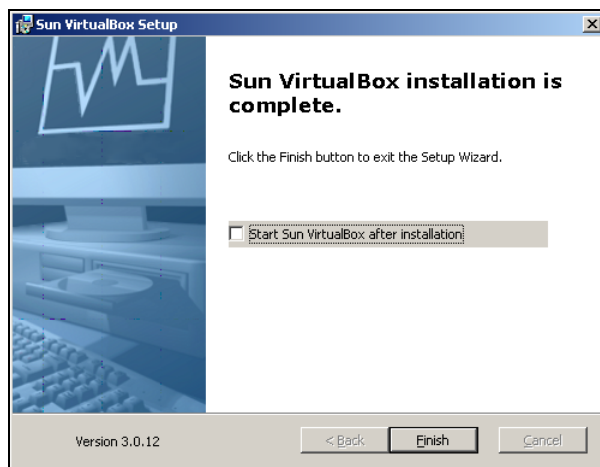
8) Windows Logo Test



You will be informed that the driver did not pass the Windows Logo Test, and asked if you want to continue⁵¹. Please answer with:

"Continue installation".

9) Installation complete



Click finish

(If you started the application with "Run as..." as Administrator, quit and uncheck "Start Oracle VirtualBox after installation").

10) Reboot

If you are asked if you want to reboot your computer answer with "Yes".

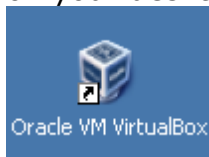
⁵¹ This window can pop up more than 10 times. Always confirm with "Continue installation".

Download of the Virtual Disc Image of BIRO

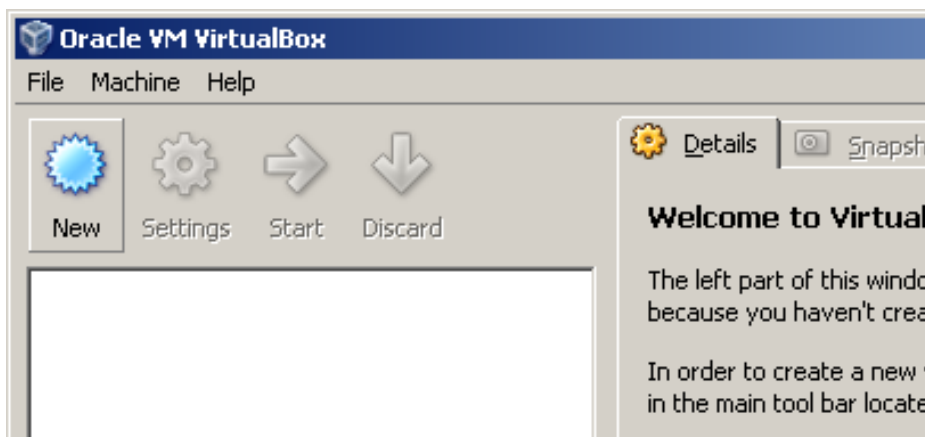
3. Direct your Web Browser to the EUBIROD webpage. Download the ZIP-File of the VirtualBox image under the download section.
4. Create a directory on a drive with enough (>4 GB) of free disk space:
e.g. d:\VirtualBox\BIROX
5. Extract the downloaded ZIP-File to this directory

Usage of BIROX in Oracle VirtualBox

Start Oracle VirtualBox by clicking on the entry in the start menu or on the link on your desktop:

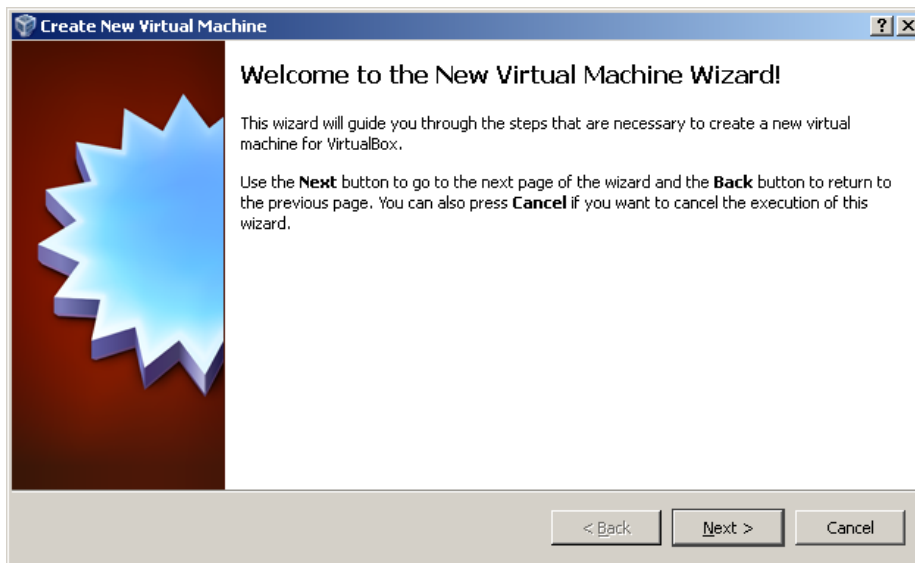


1) Create New Virtual Machine



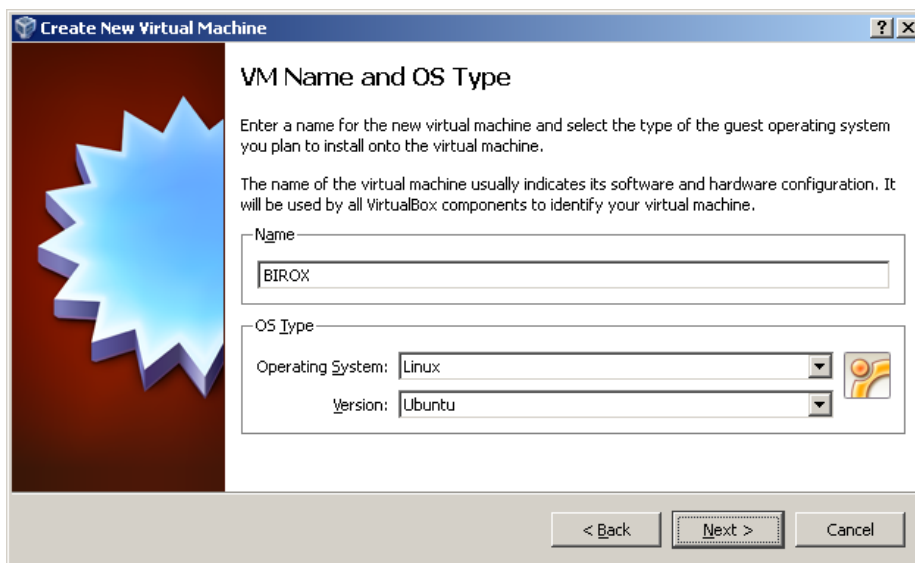
Click New

2) New Virtual Machine Wizard



Click "Next"

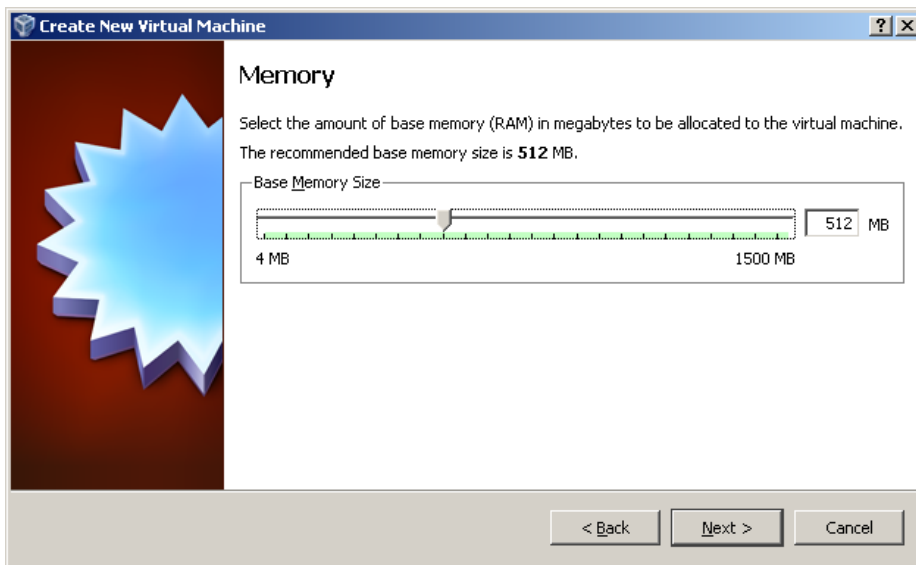
3) Create New Virtual Machine



Enter Name "BIROX"

Select "Linux" and "Ubuntu"

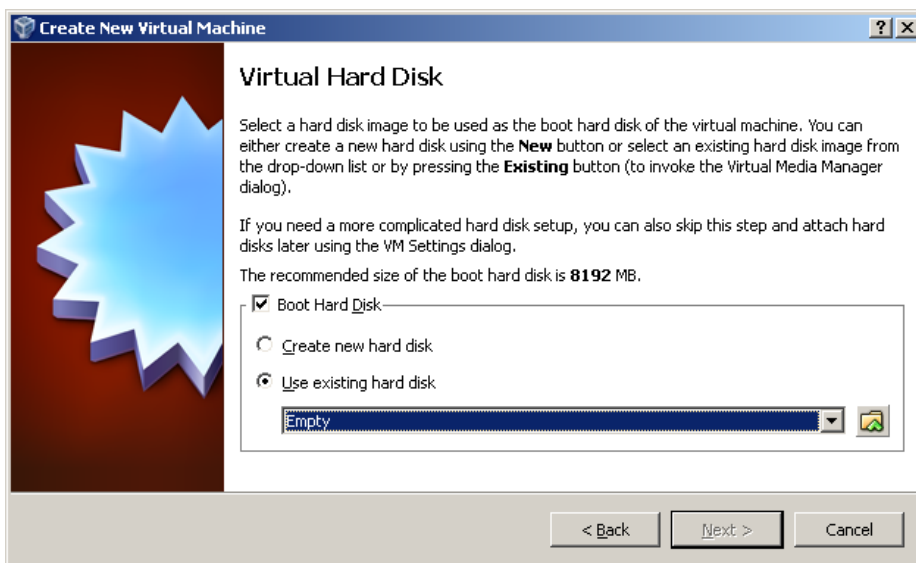
4) Set Base Memory Size



If your computer has enough memory, you can give more memory to the virtual machine.

→ See Memory Size

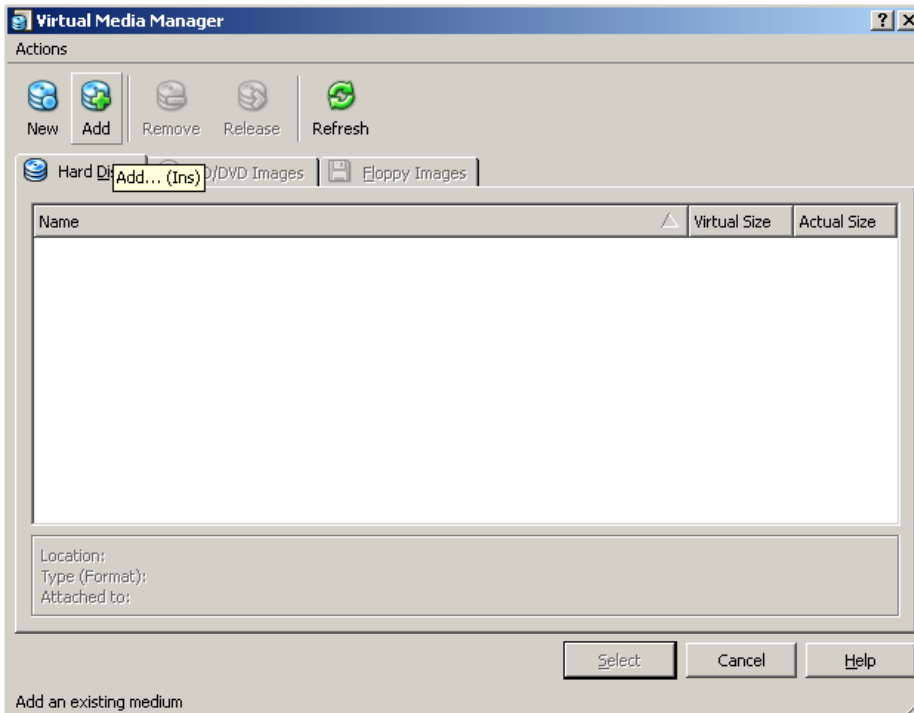
5) Hard Disk



Select "Use existing hard disk"

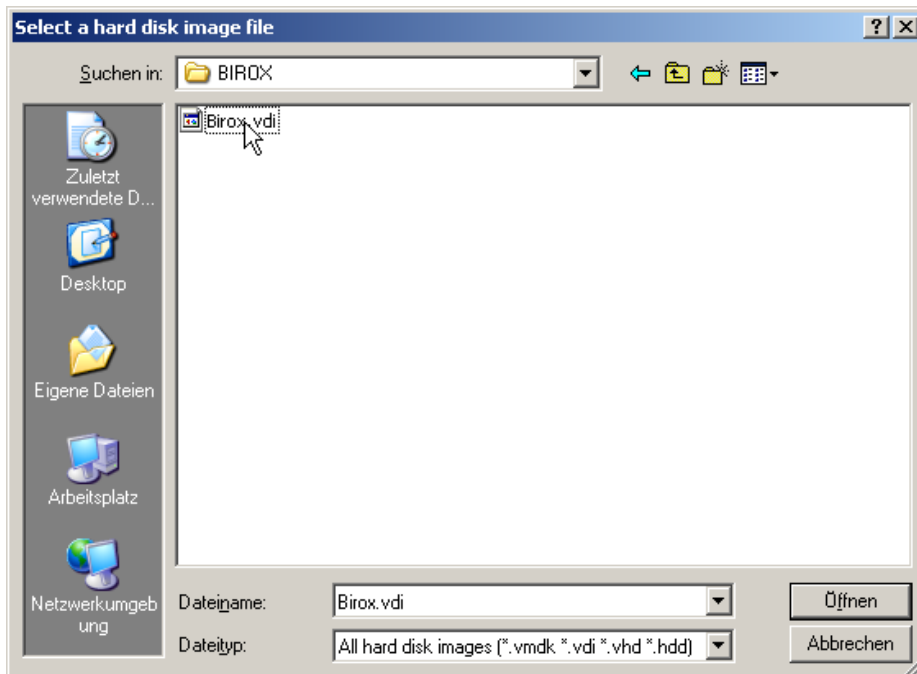
Click on the folder icon

6) Virtual Media Manager



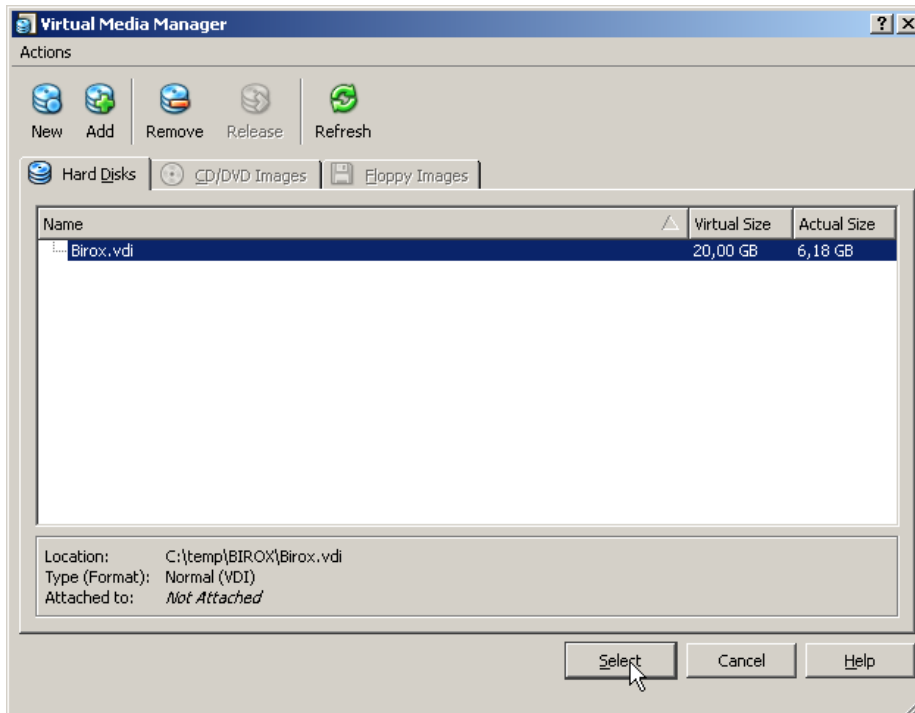
Click Add

7) Select BIRO Disk file



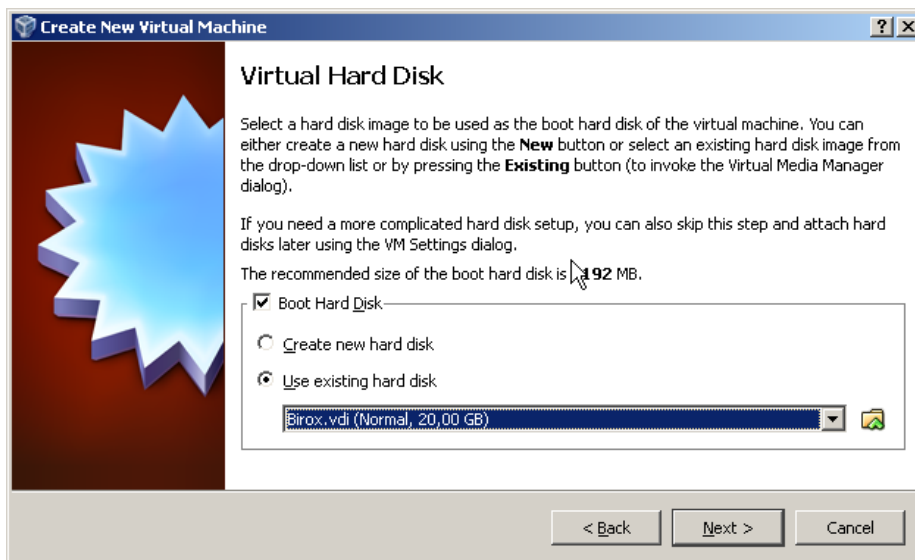
Select the "BIROX*.vdi" file that has been extracted before

8) Virtual Media Selection



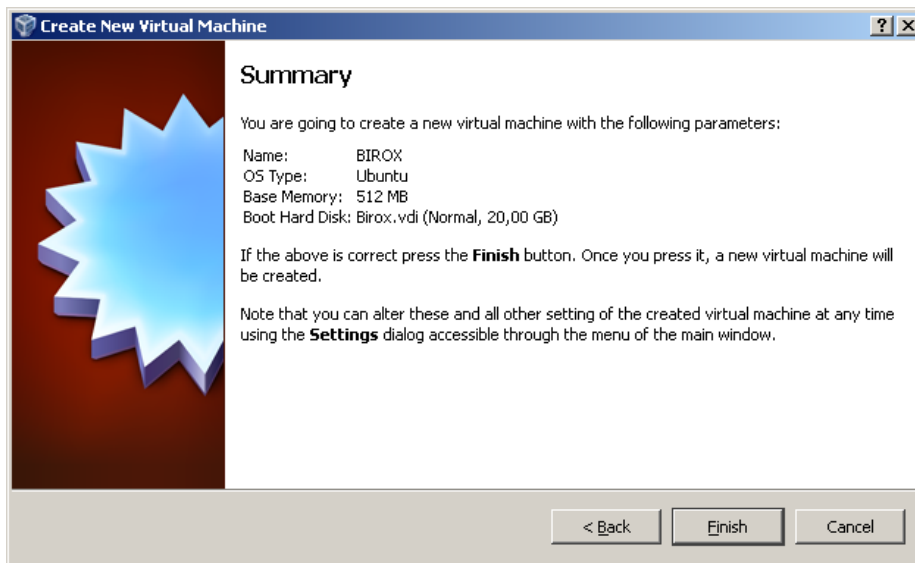
Click "Select"

9) Hard Disk selected



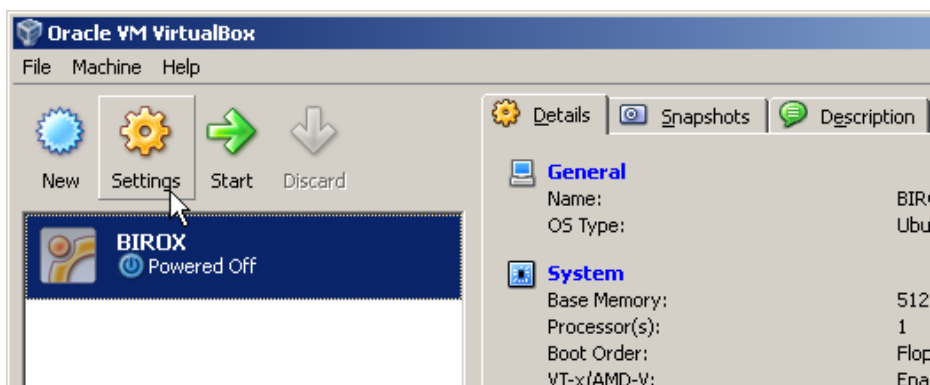
Click "Next"

10) Summary



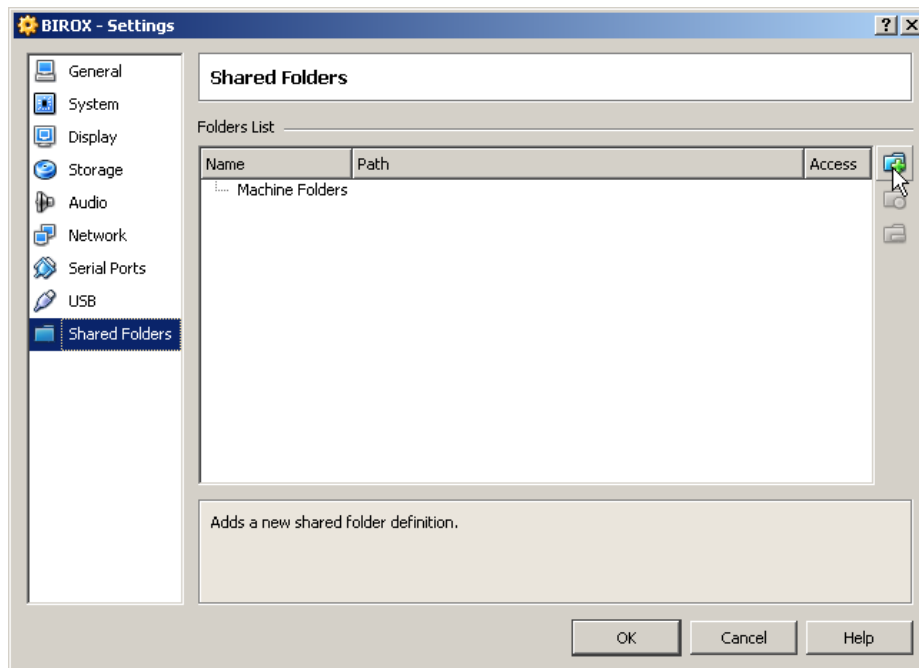
Click "Finish"

11) Edit settings



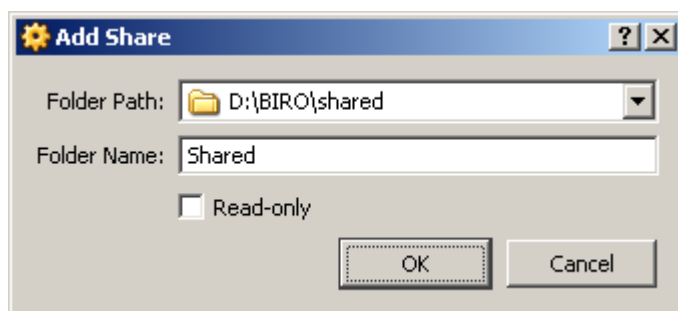
Click the "Settings" button

12) Shared Folders



Select "Shared Folders" and click "Add" on the right

13) Edit Share

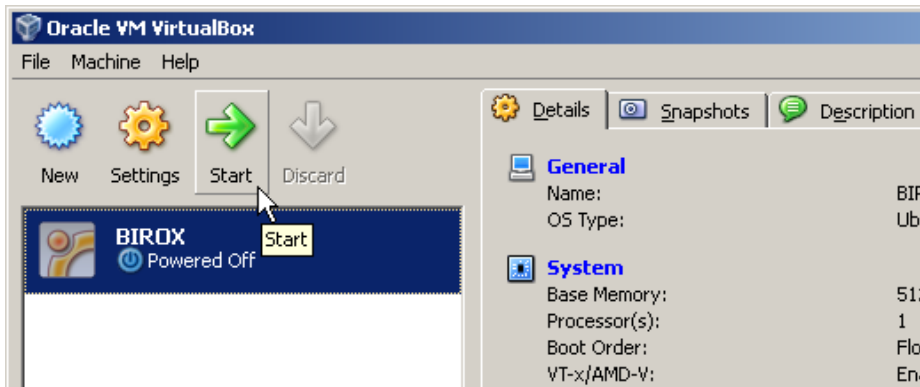


Folder Path: Set this to a directory where BIRO settings and data can remain, e.g. d:\Virtual Box\BIROX\Shared\

Folder Name: Must be set to **"Shared"**!!!!

Click "OK"

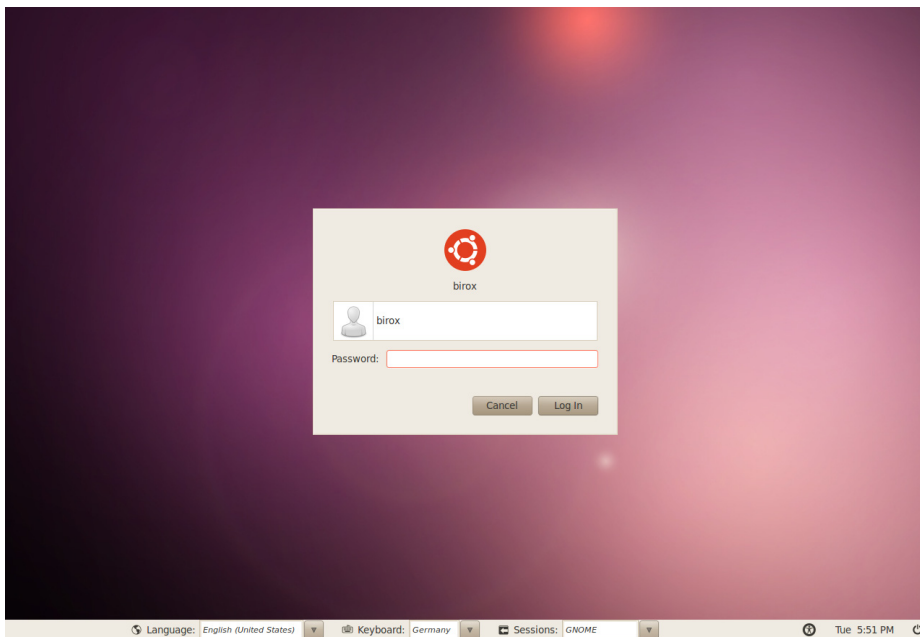
14) Start the virtual machine



Click "Start"

Now a "Computer inside your Computer" is booting. If everything works all right you will be prompted with a login.

15) Login

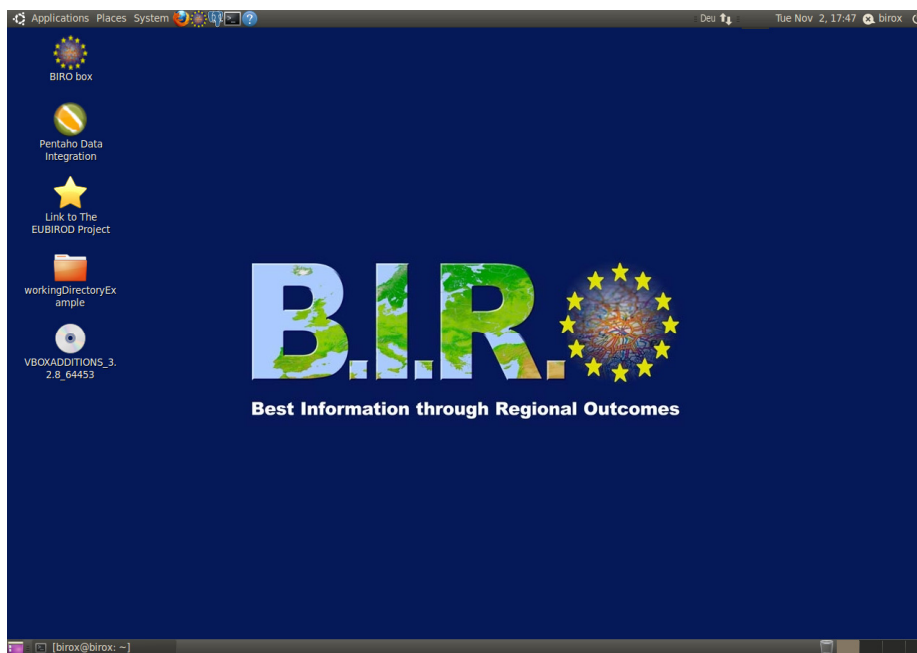


First select your keyboard layout in the bottom bar. Then login using:

User: **birox**

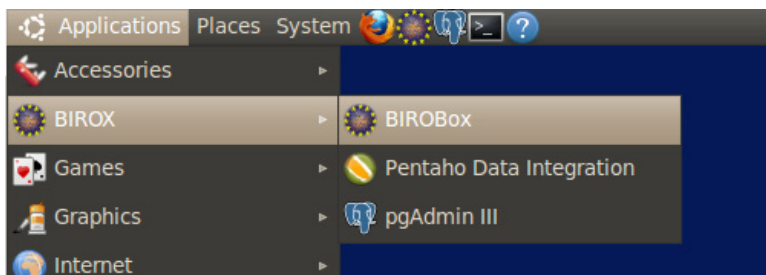
Password: **birox**

16) Start the Biro Box

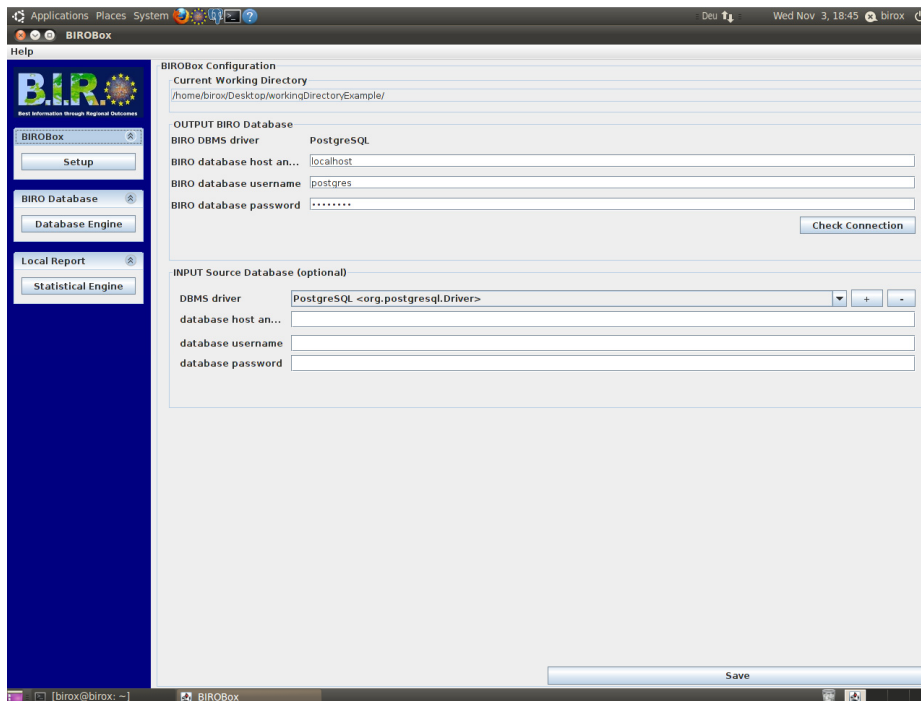


Start the Biro Box using the link on the desktop

17) You can as well use the Applications Pull-Down Menu to start the BIRO Box or other applications.



18) Use the Biro Box



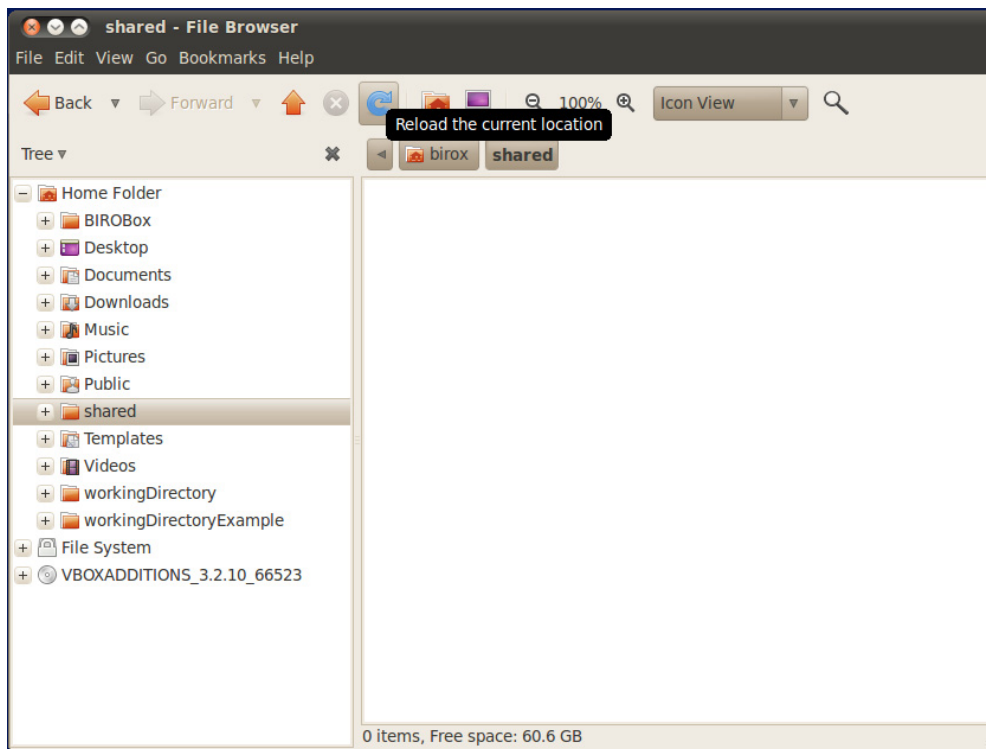
Configuration of the Exchange Directory

The Exchange Directory is a way to move files into and out of the Virtual BIROX PC.

To avoid storage of data and configuration inside the virtual BIROX, enables the possibility to easily exchange the "virtual hard-disc" with a new or modified BIROX installation. Data and configuration remain in place.

All files copied from Windows into the exchange directory will be visible in BIROX and the other way around. So data files needed by the BIROX or BIROBox can be moved in or data produced by the BIROBox can be moved out of BIROX. To see the content of the shared folder, simply open "**Places → Home Folder → shared**" in the top bar of the BIROX. A file browser pops up showing the folder.

If the **shared-Folder** cannot be opened in the BIROX, or doesn't show any content, please make sure that **steps 11 to 13** of chapter Usage of BIROX in Oracle VirtualBox have been performed during the installation of the VirtualBox image. If yes, **copy** a file on the host system (Windows,...) in this folder using a file-browser, **switch** back to BIROX and press the reload button in the file browser showing the content of the **shared**-folder:



The current versions of the BIROBox supports configuration, so that all data and settings are stored in one directory, which will be inside the exchange directory.



Simply point the path of the Working directory in the first screen of the BIROBox to your **shared Folder**, e.g. */home/birox/shared*.

Additional Information

Memory Size

You should give as much memory to your virtual machine as possible so that your host operation system is still able to run.

- Windows Vista needs 1GB
- Windows XP needs 756 MB

Examples:

- If your computer has 2 GB memory and you are running Windows Vista you can give 1GB memory to the virtual machine.
- If your computer has 1 GB memory and you are running Windows XP you can give 512 MB to the virtual machine. (That's a minimum requirement for both)
- *If your computer has 1 GB memory and you are running Windows Vista, please give 512 MB to the virtual machine and see what happens :-)*

11. Appendix B

Maven's Default Lifecycle Phases

The default lifecycle phases provide a general model of a build process for a software application. An overview of the different phases is given below.

| Lifecycle-Phase | Description |
|-------------------------|---|
| validate | Validation of the project is correct and all necessary information is available to complete a build |
| generate-sources | Generate any source code for inclusion in compilation |
| process-sources | Process the source code, for example to filter any values |
| generate-resources | Generate resources for inclusion in the package |
| process-resources | Copy and process the resources into the destination directory, ready for packaging |
| compile | Compile the source code of the project |
| process-classes | Post-process the generated files from compilation, for example to do bytecode enhancement on Java classes |
| generate-test-sources | Generate any test source code for inclusion in compilation |
| process-test-sources | Process the test source code, for example to filter any values |
| generate-test-resources | Create resources for testing |
| process-test-resources | Copy and process the resources into the test destination directory |
| test-compile | Compile the test source code into the test destination directory |

| | |
|-----------------------|--|
| test | Run tests using a suitable unit testing framework. These tests should not require the code to be packaged or deployed |
| prepare-package | Perform any operations necessary to prepare a package before the actual packaging. This often results in an unpacked, processed version of the package |
| package | Take the compiled code and package it in its distributable format, such as a JAR, WAR, EJB, POM or EAR |
| pre-integration-test | Perform actions required before integration tests are executed. This may involve things as setting up the required environment |
| integration-test | Process and deploy the package if necessary into an environment where integration tests can be run |
| post-integration-test | Perform actions required after integration tests have been executed. This may include cleaning up the environment |
| verify | Run any checks to verify the package is valid and meets quality criteria |
| install | Install the package into the local repository, for use as a dependency in other projects locally |
| deploy | Copies the final package to the remote repository for sharing with other developers and projects (usually only relevant during a formal release) |

Table 2 – Maven Lifecycle Phases^{vii}

12. Appendix C

Technical Infrastructure Questionnaire

Survey about Technical Infrastructure and Data Sources in EUBIROD

Deadline: Please submit this questionnaire for your institution to the EUBIROD Coordination Centre (EUBIROD@unipg.it) within 26th April 2009.

This questionnaire is sent to you as a starting point for Work Package 7 "Technology Transfer" in the EUBIROD project by Joanneum Research, Austria.

Institution Name: Institute of Health Management, Joanneum Research, Austria

Technical Contact Person

Who may be contacted in your institution concerning technical issues?

Name: _____

Email: _____

Phone: _____

Sample Data Set

We kindly ask you to **send us a sample data set** with 2-3 sample data entries in an export format you currently use to give us an impression of the required transformations.

Data Source Information

- Please **fill in the following data** on the data sources your institution will provide in EUBIROD.
- Please note that in EUBIROD we assume that each partner already has a "local data repository" (register...), where data from individual diabetes clinics or other sources of information is collected. The questions here relate to the data and availability of this local data repository.
- If you can provide more than one data source, please use the columns for 2nd and 3rd data source.

| 1 st Data Source | 2 nd Data Source | 3 rd Data Source |
|---|---|---|
| Data Source Short Name: | Data Source Short Name: | Data Source Short Name: |
| a) What type of data is available in your local repository? <input type="checkbox"/> Clinical data containing episode data (e.g. multiple lab results per year) <input type="checkbox"/> Clinical data collected | a) What type of data is available in your local repository? <input type="checkbox"/> Clinical data containing episode data (e.g. multiple lab results per year) <input type="checkbox"/> Clinical data collected | a) What type of data is available in your local repository? <input type="checkbox"/> Clinical data containing episode data (e.g. multiple lab results per year) <input type="checkbox"/> Clinical data collected |

| | | |
|---|---|---|
| periodically (e.g. DiabCare forms) <input type="checkbox"/> Health Survey (no continuous data collection) <input type="checkbox"/> other: General Comment: | periodically (e.g. DiabCare forms) <input type="checkbox"/> Health Survey (no continuous data collection) <input type="checkbox"/> other: General Comment: | periodically (e.g. DiabCare forms) <input type="checkbox"/> Health Survey (no continuous data collection) <input type="checkbox"/> other: General Comment: |
| b) In which form is the data available? <input type="checkbox"/> Database: <input type="checkbox"/> Text File(s) <input type="checkbox"/> Excel File(s) <input type="checkbox"/> XML File(s) <input type="checkbox"/> Statistical Software data Format (SPSS, SAS...) <input type="checkbox"/> other: General Comment: | b) In which form is the data available? <input type="checkbox"/> Database: <input type="checkbox"/> Text File(s) <input type="checkbox"/> Excel File(s) <input type="checkbox"/> XML File(s) <input type="checkbox"/> Statistical Software data Format (SPSS, SAS...) <input type="checkbox"/> other: General Comment: | b) In which form is the data available? <input type="checkbox"/> Database: <input type="checkbox"/> Text File(s) <input type="checkbox"/> Excel File(s) <input type="checkbox"/> XML File(s) <input type="checkbox"/> Statistical Software data Format (SPSS, SAS...) <input type="checkbox"/> other: General Comment: |
| c) Which aggregation level does the data source provide? <input type="checkbox"/> Episode data (several data sets per patient per year or quarter) <input type="checkbox"/> Individual patient data (one data set per patient per year or quarter) <input type="checkbox"/> Data is available pre-aggregated per <input type="checkbox"/> Region (e.g. State, County, District, ...) Please Specify: <input type="checkbox"/> Time (e.g. Year, Quarter, ...) Please Specify: <input type="checkbox"/> Other: General Comment: | c) Which aggregation level does the data source provide? <input type="checkbox"/> Episode data (several data sets per patient per year or quarter) <input type="checkbox"/> Individual patient data (one data set per patient per year or quarter) <input type="checkbox"/> Data is available pre-aggregated per <input type="checkbox"/> Region (e.g. State, County, District, ...) Please Specify: <input type="checkbox"/> Time (e.g. Year, Quarter, ...) Please Specify: <input type="checkbox"/> Other: General Comment: | c) Which aggregation level does the data source provide? <input type="checkbox"/> Episode data (several data sets per patient per year or quarter) <input type="checkbox"/> Individual patient data (one data set per patient per year or quarter) <input type="checkbox"/> Data is available pre-aggregated per <input type="checkbox"/> Region (e.g. State, County, District, ...) Please Specify: <input type="checkbox"/> Time (e.g. Year, Quarter, ...) Please Specify: <input type="checkbox"/> Other: General Comment: |
| d) Which activities are you (or others) already performing with this data regionally? <input type="checkbox"/> Health reporting <input type="checkbox"/> Epidemiology and other scientific work <input type="checkbox"/> Benchmarking health care providers <input type="checkbox"/> Benchmarking regions <input type="checkbox"/> Health Services Planning <input type="checkbox"/> Health Services Monitoring <input type="checkbox"/> Accounting, refunding and | d) Which activities are you (or others) already performing with this data regionally? <input type="checkbox"/> Health reporting <input type="checkbox"/> Epidemiology and other scientific work <input type="checkbox"/> Benchmarking health care providers <input type="checkbox"/> Benchmarking regions <input type="checkbox"/> Health Services Planning <input type="checkbox"/> Health Services Monitoring <input type="checkbox"/> Accounting, refunding and | d) Which activities are you (or others) already performing with this data regionally? <input type="checkbox"/> Health reporting <input type="checkbox"/> Epidemiology and other scientific work <input type="checkbox"/> Benchmarking health care providers <input type="checkbox"/> Benchmarking regions <input type="checkbox"/> Health Services Planning <input type="checkbox"/> Health Services Monitoring <input type="checkbox"/> Accounting, refunding and |

| | | |
|---|---|---|
| other administrative processes <input type="checkbox"/> other, please specify: General Comment: | other administrative processes <input type="checkbox"/> other, please specify: General Comment: | other administrative processes <input type="checkbox"/> other, please specify: General Comment: |
| e) Will your Institution be able to provide the merge table, in which format? <input type="checkbox"/> yes, as database: <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | e) Will your Institution be able to provide the merge table, in which format? <input type="checkbox"/> yes, as database: <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | e) Will your Institution be able to provide the merge table, in which format? <input type="checkbox"/> yes, as database: <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: |
| f) Will your Institution be able to provide the activity table, in which format? <input type="checkbox"/> yes, as database: <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | f) Will your Institution be able to provide the activity table, in which format? <input type="checkbox"/> yes, as database: <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | f) Will your Institution be able to provide the activity table, in which format? <input type="checkbox"/> yes, as database: <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: |
| g) Will your Institution be able to provide the population table? <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | g) Will your Institution be able to provide the population table? <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | g) Will your Institution be able to provide the population table? <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: |
| h) Will your Institution be able to provide the diabetic population table? <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | h) Will your Institution be able to provide the diabetic population table? <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: | h) Will your Institution be able to provide the diabetic population table? <input type="checkbox"/> yes, as CSV file <input type="checkbox"/> yes, with a different format: <input type="checkbox"/> no General Comment: |
| i) Will your Institution be able to provide information on site | i) Will your Institution be able to provide information on site | i) Will your Institution be able to provide information on site |

| | | |
|--|--|--|
| profile? <input type="checkbox"/> yes <input type="checkbox"/> no General Comment: | profile? <input type="checkbox"/> yes <input type="checkbox"/> no General Comment: | profile? <input type="checkbox"/> yes <input type="checkbox"/> no General Comment: |
|--|--|--|

13. Appendix D

Technical Infrastructure Summary

This table summarizes the responses of the participating countries in the EUBIROD project to the questionnaire as seen in Appendix C
Technical Infrastructure Questionnaire

| | Data Source Short Name | Type of data? | Format? | Notes | Aggregation level? | Performed activities on dataset? | Merge table? | Notes | Activity table? | Notes | Population table? | Notes | Diabetic population table? | Notes | Site profile? |
|----------------|---------------------------|------------------------------|----------|---------------------|----------------------------|---|-----------------|------------------|--------------------|-------|----------------------|-------|----------------------------------|-------|------------------|
| Austria | FQSDÖ | CD collected periodically | database | MS SQL 2000 | individual patient data | Epidemiology and other , Accounting, refunding and other administrative processes | yes | View in MsSQL | no | | yes | csv | no | | yes |
| Belgium | IKED_EUBIROD | CD collected periodically | database | dta | individual patient data | health reporting, epidemiology and scientific work, benchmarking HCP | yes | csv | no | | yes | csv | no | | no |
| Croatia | CroDiab | CD for episode data | database | | individual patient data | health reporting, epidemiology and scientific work | yes | csv | yes | csv | yes | csv | yes | csv | yes |
| Cyprus | - | CD for episode data | database | access, textfile | episode data | health reporting | yes | View in MsSql | no | | no | - | no | | yes |

EUBIROD Project - DELIVERABLE D7.1 - Customized Toolbox

| | Data Source Short Name | Type of data? | Format? | Notes | Aggregation level? | Performed activities on dataset? | Merge table? | Notes | Activity table? | Notes | Population table? | Notes | Diabetic population table? | Notes | Site profile? |
|---------|---|--|-------------------------------------|----------------|----------------------------|--|-----------------|-------|--------------------|-------|----------------------|-------|----------------------------------|-------|------------------|
| Denmark | HUH | CD collected periodically | excel | | individual patient data | health reporting, benchmarking health care providers, benchmarking regions | yes | csv | yes | csv | yes | csv | yes | csv | yes |
| Germany | FQSDD | CD collected periodically | database | MS SQL 2000 | individual patient data | benchmarking HCP, benchmarking regions, benchmarking centres | yes | csv | yes | csv | yes | csv | yes | csv | yes |
| Hungary | Follow-Up Study | health survey(no continuous data collection) | text, excel, xml, spps/sas | | individual patient data | Health reporting, Epidemiology and other scientific work, Benchmarking health care providers, Benchmarking regions, Health Services Planning, Health Services Monitoring | yes | csv | no | | yes | csv | yes | csv | yes |
| | General Practitioners' Morbidity Sentinel Stations Programme | health monitoring in primary care | text, excel, xml, spps/sas | | episode data | Health reporting, Epidemiology and other scientific work, Health Services Planning, Health Services Monitoring | yes | csv | no | | yes | csv | yes | csv | yes |

| | Data Source Short Name | Type of data? | Format? | Notes | Aggregation level? | Performed activities on dataset? | Merge table? | Notes | Activity table? | Notes | Population table? | Notes | Diabetic population table? | Notes | Site profile? |
|-------------------------|---------------------------|--|-------------------------------------|-------|----------------------------|--|-----------------|------------|--------------------|----------------|----------------------|----------------|----------------------------------|----------------|------------------|
| | Cross-sectional study | health survey(no continuous data collection) | text, excel, xml, spps/sas | | individual patient data | Health reporting, Epidemiology and other scientific work, Benchmarking health care providers, Benchmarking regions, Health Services Planning, Health Services Monitoring | yes | csv | no | | yes | csv | yes | csv | yes |
| Ireland | Diamond | CD for episode data | text, excel | | episode data | Epidemiologic and other scientific work, Health Services Planning | yes | csv | no | | no | | yes | csv | yes |
| Italy | Data not available | | | | | | | | | | | | | | |
| Kuwait | Data not available | | | | | | | | | | | | | | |
| Luxem- bourg | Diab CNS | national reimbursement | SPSS, SAS | | episode data | Epidemiology and other scientific work | yes | database | no | | yes | csv | yes | SAS | yes |
| Malta | Data not available | | | | | | | | | | | | | | |
| Netherlands | Hoorndata | CD collected periodically | excel, SPSS/SAS | | individual patient data | Health reporting, Epidemiology and other scientific work, Benchmarking health care providers, Health Services Planning, Health Services | yes | excel/SPSS | yes | excel/ SPSS | yes | excel/ SPSS | yes | excel/ SPSS | yes |

EUBIROD Project - DELIVERABLE D7.1 - Customized Toolbox

| | Data Source Short Name | Type of data? | Format? | Notes | Aggregation level? | Performed activities on dataset? | Merge table? | Notes | Activity table? | Notes | Population table? | Notes | Diabetic population table? | Notes | Site profile? |
|----------|--|------------------------------|----------|--------------------------------------|----------------------------|--|-----------------|-------|--------------------|-------|----------------------|-------|----------------------------------|-------|------------------|
| | | | | | | Monitoring | | | | | | | | | |
| Norway | Norwegian Diabetes register (Current) | CD collected periodically | database | ORACLE | individual patient data | None at the moment. | no | - | no | | no | | no | | no |
| Poland | Silesia | CD for episode data | database | MS SQL, text, excel, xml | episode data | Epidemiology and other scientific work | yes | csv | no | | yes | csv | no | | yes |
| Romania | DiabCare | CD collected periodically | database | MySQL | individual patient data | Epidemiology and other scientific work, Benchmarking health care providers, Benchmarking regions | yes | MySQL | no | | no | | no | | no |
| | paulescu | CD for episode data | excel | | episode data | Health reporting, Accounting, refunding and other administrative processes | yes | csv | no | | yes | csv | no | | yes |
| | telediab | CD for episode data | database | MySQL, text | episode data | Health reporting, Epidemiology and other scientific work | yes | csv | no | | yes | csv | no | | yes |
| Scotland | Data not available | | | | | | | | | | | | | | |
| Slovenia | Data not available | | | | | | | | | | | | | | |

EUBIROD Project - DELIVERABLE D7.1 - Customized Toolbox

| | Data Source Short Name | Type of data? | Format? | Notes | Aggregation level? | Performed activities on dataset? | Merge table? | Notes | Activity table? | Notes | Population table? | Notes | Diabetic population table? | Notes | Site profile? |
|--------|---------------------------|---|---------|-------|---|---|-----------------|-----------|--------------------|-------|----------------------|-------|----------------------------------|-------|------------------|
| Spain | Data not available | | | | | | | | | | | | | | |
| Sweden | SPCD | CD for episode data, CD collected periodically | excel | | episode data, individual patient data | Health reporting, Epidemiology and other, Health Services Planning, Health Services Monitoring | yes | MS Access | no | | yes | excel | yes | excel | yes |

CD.....clinical data

csv.....comma separated values

14. References

-
- ⁱ The BIRO Consortium (2009). *Best Information through Regional Outcomes: a Shared European Diabetes Information System for Policy and Practice*. The BIRO Consortium, European Commission:
http://www.eubirod.eu/documents/downloads/BIRO_Monograph.pdf
- ⁱⁱ *Council conclusions on promotion of healthy lifestyles and prevention of Type 2 diabetes (2006/C 147/01)*; available at: <http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:C:2006:147:0001:0004:EN:PDF>
- ⁱⁱⁱ The EUBIROD Consortium (2009). EUBIROD WP4: Deliverable 4.1 - *Report on Training*
- ^{iv} Benington, H.D. (1956). *Production of large computer programs*. In Proceedings ONR Symposium on Advanced Programming Methods for Digital Computers, 15-27.
- ^v Royce, W. W. (1970). *Managing the development of large software systems*. In Proceedings IEEE Wescon, 1-9.
- ^{vi} Ruparelia, N.B. (2010). *Software Development Lifecycle Models*. In ACM Sigsoft Software Engineering Notes, 35(3), 8-13
- ^{vii} O'Brien, T. et. al. (2010). *Maven: The complete reference*. Mountain View, CA: Sonatype.
- ^{viii} Carzaniga, A. et. al. (1998). *A Characterization Framework for Software Deployment Technologies*. Tech. Report CU-CS-857-98, Dept. of Computer Science, University of Colorado.
- ^{ix} Rodin, J. (2010). *Debian New Maintainers' Guide*. From www.debian.org/doc/manuals/maint-guide/maint-guide.en.pdf on 23rd of August 2010
- ^x The BIRO Consortium (2006). *BIRO WP3: Deliverable 3.1 - Common Dataset*
- ^{xi} The EUBIROD Consortium (2010). *EUBIROD WP5: Deliverable 5.1 - Common Dataset*
- ^{xii} Bouman, R., & van Dongen, J. (2009). *Pentaho Solutions: Business Intelligence and Data Warehousing with Pentaho and MySQL*. Indianapolis, IN: Wiley.