University of Belgrade
Faculty of Organizational Sciences
Center for Business Decision Making

WHIBO Developer guide

Belgrade, November 2013

Introduction

WhiBo is a RapidMiner (Mierswa et al. 2006) plug-in for "white-box" component based design of decision tree algorithms for classification and evaluation of these algorithms and their parts. It is intended to be used by typical end users, research scientists and algorithm developers. The main idea of WhiBo is to offer standardized components for algorithm design which will enable simple design and performance testing, easy extension of the component repository and creation of new generic algorithms. Currently, WhiBo provides one generic algorithm, a graphical interface and a component repository for design of decision trees for classification. A framework for performance testing is implemented in WhiBo as well. WhBo plug-in and source code, is available from www.whibo.fon.rs. Source code is documented thoroughly and accessible from the web site through the API documentation. The web site also provides installation guide and number of tutorials for end users, algorithm developers, and research scientists.

Black-box approach

Data mining algorithms are usually implemented in a "black-box" manner. This means that the user defines input data and parameters (if needed) for the algorithm, and the algorithm produces a model. The user has no other possibilities to modify the algorithm to better adjust to data. The "black-box" approach is satisfying for most users. On the other hand, implementation of algorithms as a "black-box" makes it more difficult for algorithm designers who want to use parts of the existing algorithm to create new algorithms. The structure of black box algorithms demands reimplementation of algorithms and their parts from the scratch. "Black-box" implemented algorithms are harder to evaluate and analyze, because it is not clear which part of the algorithm has influence on overall algorithm performance.

White-Box approach

The “white-box” approach allows the user to define parameters, and inputs (as in black-box algorithms) of an algorithm, but also the building blocks (i.e. components) of the algorithm. These components are solutions for typical sub-problems consistently encountered in the process of constructing the appropriate model for the data at hand. This way, algorithmic solution becomes more data and user driven, since it enables the users to intelligently select components of the algorithm which best address the problems of the specific data. Moreover, good ideas from algorithms are saved within components, so they can be used in other algorithms.

White-box approach offers several advances in comparison with black box algorithms (Sonnenburg et al, 2007).

- Combining advantages of various algorithms,
- Comparing algorithms in more details,
- Building on existing resources with less re-implementation,
- Easier “bug” detection on the level of components,
- Collaborative emergence of standards.

WhiBo component repository and Generic decision tree (GDT) algorithm

WhiBo includes a reusable component repository for design of decision tree algorithms. These components were extracted from “black-box” algorithms:

- ID3 (Quinlan JR, 1986),
- C4.5 (Quinlan JR, 1993),
- CART (Breiman et al, 1984),
- CHAID (Kass GV, 1980)

and improvements (distance measure identified in (Mantaras, 1991). Description of analyzed algorithms and partial improvements could be found in Appendix A.

Sub-problems and solutions (reusable components)

In WhiBo algorithms are built by choosing building blocks (i.e. reusable components - RCs) for each sub-problem. The problem of building decision tree model is divided into sub-problems that are generalized algorithm structures with the same input and output structure identified in all analyzed algorithms. Every sub-problem with defined inputs and outputs can be solved in many ways, i.e. with various a reusable components (RCs). That means that every RC solves a specific sub-problem which has the same I/O.

Table 1 shows identified sub-problems and components with their corresponding I/O that are currently implemented in WhiBo.

<table>
<tr><th>Sub-problem</th><th>Reusable component</th><th>Input</th><th>Output</th></tr>
<tr><td>Remove insignificant attributes</td><td>F TEST (numerical attributes)
CHI SQUARE TEST (categorical attributes)</td><td>Dataset in current node</td><td>Dataset in current node (reduced)</td></tr>
<tr><td>Create split (Numerical)</td><td>BINARY</td><td rowspan="4">Dataset in current node</td><td rowspan="4">A split candidate</td></tr>
<tr><td rowspan="3">Create split (Categorical)</td><td>BINARY</td></tr>
<tr><td>MULTIWAY</td></tr>
<tr><td>SIGNIFICANT</td></tr>
<tr><td rowspan="5">Evaluate split</td><td>CHI SQUARE</td><td rowspan="5">A split candidate</td><td rowspan="5">The best split in current node</td></tr>
<tr><td>INFORMATION GAIN</td></tr>
<tr><td>GAIN RATIO</td></tr>
<tr><td>GINI</td></tr>
<tr><td>DISTANCE MEASURE</td></tr>
<tr><td rowspan="2">Stop criteria</td><td>MAXIMAL TREE DEPTH</td><td rowspan="2">Current tree model</td><td rowspan="2">Signal for stopping tree growth in current node</td></tr>
<tr><td>MINIMAL NODE SIZE</td></tr>
<tr><td rowspan="2">Prune tree</td><td>PESSIMISTIC ERROR PRUNING (PEP)</td><td rowspan="2">Current tree model</td><td rowspan="2">Pruned tree model</td></tr>
<tr><td>MIN LEAF SIZE (MLS)</td></tr>
</table>

Table 1 - Sub-problems, reusable components with standardized I/O for Generic decision tree algorithm

Sub-problems and reusable components implemented in Whibo are described according to Tracz (1990) in Appendix B.

Generic decision tree (GDT) structure

The GDT structure proposed in WhiBo is shown on Figure 1. For sub-problems that are bolded it is necessary to define a sub-problem, while for other sub-problems RCs are optional to use. “Create split” (numerical, and categorical) and “Evaluate split” RCs are necessary for decision tree growth. Besides that, there are no restrictions for combinations of RCs.

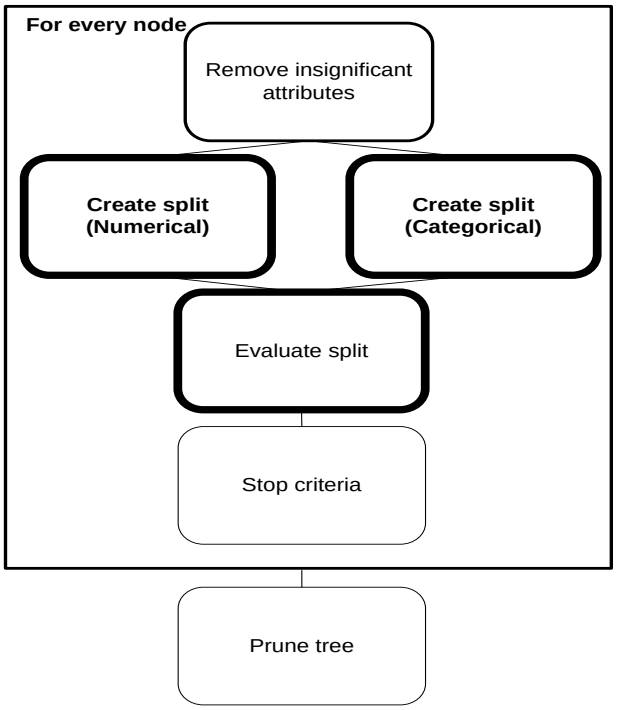

Figure 1 - Generic decision tree (GDT) algorithm

The proposed GDT structure and component repository enables:

- Reconstruction of the original algorithms in the parts that were analyzed.
- Creation of hybrid algorithms with components.
- Extension of the component repository by analyzing new algorithms or partial improvements which can be incorporated in sub-problems with the same input-output structure.
- Definition of new sub-problems which can be incorporated in GDT structure.

Extending WHIBO

Input and output are well defined for every Sub-problem, and these sub-problems are implemented as abstract classes in WhiBo. Reusable components are concrete classes where the logic is implemented. Sub-problems define standardized input and output for every reusable component, extended from sub-problem.

WhiBo is implemented as an extendable environment in the Java programming language that enables the implementation of new RCs and sub-problems. Extending the GDT can be done by:

- Adding new RCs.
- Adding new sub-problems in the existing GDT algorithm.

The GDT algorithm is implemented independently of RCs. So extending the GDT algorithm with a new RC asks for no changes in the algorithm flow. On the other hand, when extending WhiBo with a new sub-problem changes are needed in the GDT algorithm.

Adding a new RC is accomplished in two steps. The first step is to define a new class for the RC. For that class the user has to define parameters and implement the RC logic. The inputs and outputs of the RC are predefined by the sub-problem the RC belongs to. The necessary changes are shown in bold at Figure 55. The second step is to register the new RC for a sub-problem as shown in bold at Figure 56.

If these two steps are done correctly the user will see his own component in the central panel of WhiBo GUI (Figure 4), and can use the GDT with the new RC.

Adding a new sub-problem is achieved in three steps. The first step is to create an interface for the sub-problem, and define inputs and outputs for the sub-problem as shown in bold at Figure 57.
The second step is to register the new sub-problem to enable using it through GUI as shown in bold at Figure 58.

Finally, the user has to modify the existing GDT algorithm to utilize the newly defined sub-problem. WhiBo is not only intended for use with decision-tree algorithms, but can be extended to other component-based machine learning algorithms.

```
package rs.fon.WhiBo.GDT.component.splitEvaluation;
public class MySplitEvaluation
   extends AbstractSplitEvaluation {

@Parameter(defaultValue="0.05", minValue ="0",
                maxValue="1")
    private Double Alpha_Value;

 @Override
   public        double        evaluate(SplittedExampleSet
exampleSet)
          {
                /*
                  user implementation
                  for candidate split evaluation
                */
             return splitEvaluation;
           }
}
```

Figure 2 - Implementing a new RC

```
public class SplitEvaluation implements Subproblem {
            …
     PrivateString[] availableImplementationClassNames
=
            {
          GainRatio.class.getName(),
          GiniIndex.class.getName(),
          InformationGain.class.getName(),
          DistanceMeasure.class.getName(),
          ChiSquare_FTest.class.getName(),
          MySplitEvaluation.class.getName();
          }
};
```

Figure 3 - Registering the new RC for a sub-problem

```
package
rs.fon.WhiBo.GDT.component.newSubproblem;
public interface newSubproblem {
    public output1 newSubproblemMethod1(inputs1);
    public output2 newSubproblemMethod1(inputs2);
}
```

Figure 4 - Defining a new sub-problem

WhiBo can be found at the following web page http://code.google.com/p/WhiBo/. Data mining and machine learning researchers are invited to join our efforts to exchange components of decision trees and other machine learning algorithms in an open way based on the proposed WhiBo platform, as to establish a standard for interchange of components among decision tree based classification algorithms, as well as other machine learning algorithms.

```
Package  rs.fon.WhiBo.GDT.problem;
....
public class GenericTreeProblemBuilder {

    public Problem buildProcess() {
        …
        Subproblem s2 = new PossibleSplit();
        Subproblem s3 = new Split Evaluation();
        …
        Subproblem s7 = new UserDefinedSubproblem();
        List”Subproblem” subproblems;
        subproblems.add(s1);
        subproblems.add(s2);
        …
        subproblems.add(s7);
        Problem process = new GenericTreeProblem();
        process.setProcessSteps(steps);
        return process;
    }
....
}
```

Figure 5 - Registering the new sub-problem

Developer guide

In order to extend WhiBo there are several steps which needs to be done.

1. Since WhiBo is written in Java programming language, first step is to download Eclipse (http://www.eclipse.org/downloads/).

2. When Eclipse is downloaded subversion support needs to be installed. We recommend **Subclipse**, which can be found on http://subclipse.tigris.org/servlets/ProjectProcess?pageID=p4wYuA. Installation of **Subclipse** is done in several steps:

    1. Open **Eclipse**.
    2. Select the **Help > Install New Software** menu option.

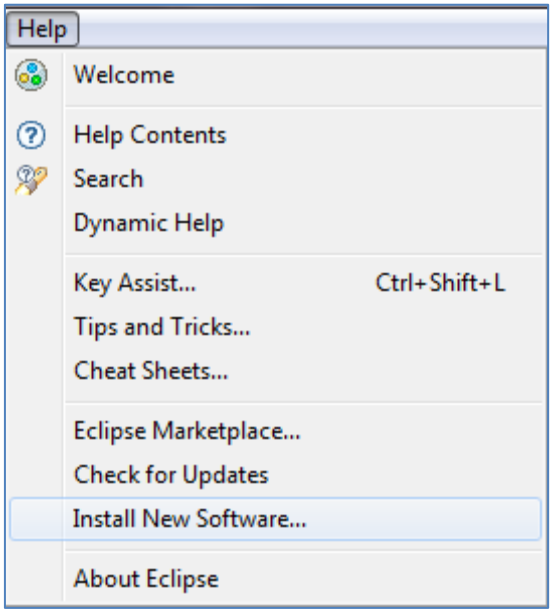

Figure 6 - Installation of Subclipse

    3. Click the **Add** button and set the **Location** field on http://subclipse.tigris.org/update_1.8.x**, and set name for example Subclipse. Then click OK button.**

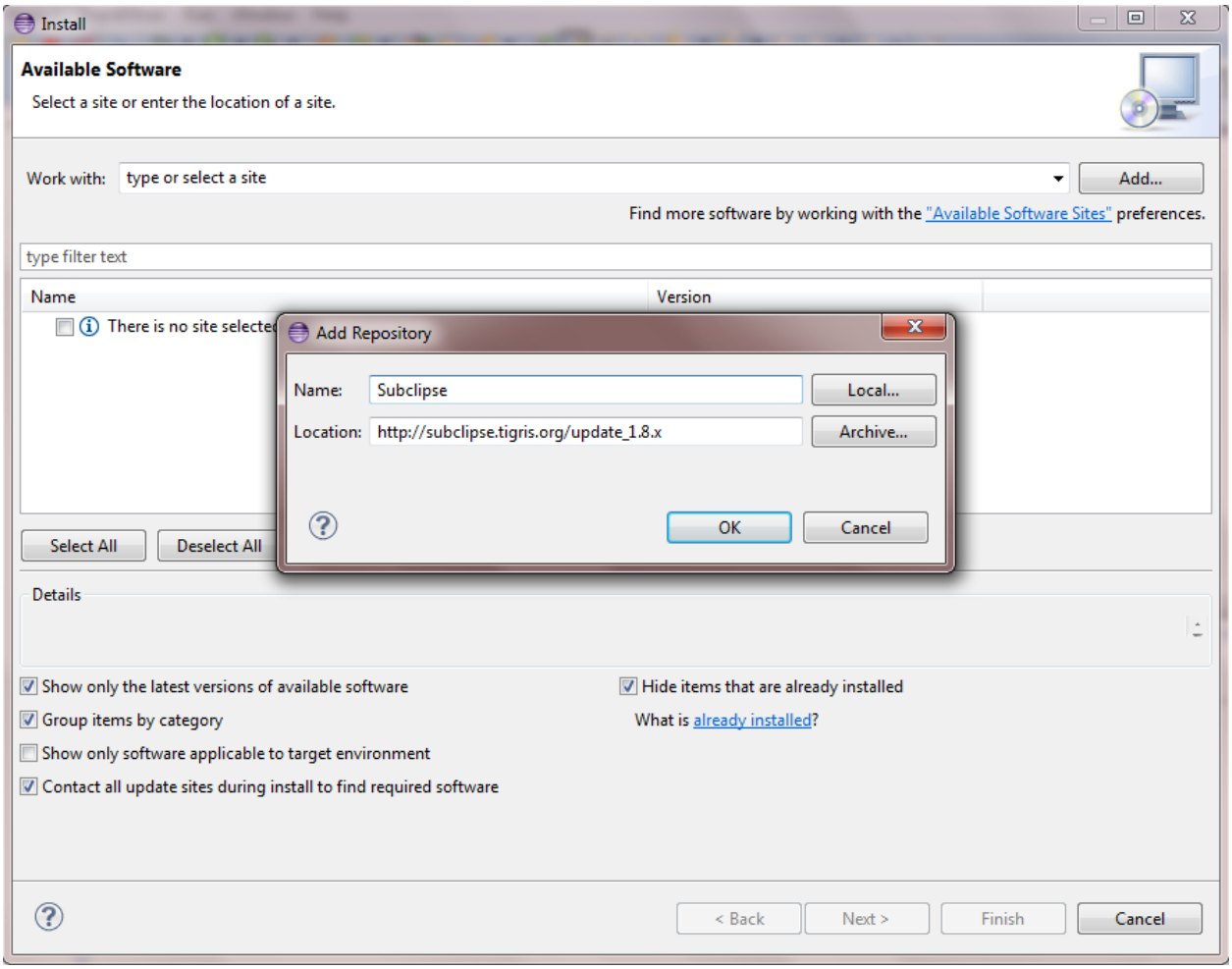

Figure 7 - Adding Subclipse repository

4. Select **Subclipse** components and click **Next**.
5. Select the **I accept the terms of the license agreements** radio button.
6. Click the **Finish** button.
7. Click **Yes** to restart Eclipse.

Eclipse will now have SVN Repository Exploring panel. If Eclipse don't show this panel at first it can be added by clicking **Windows->Open Perspective->Other…**, then selecting **SVN Repository Exploring** option and click OK.

3. Checkout of **WhiBo** project is done in several steps:

    1. Right Click a repository in the **SVN Repositories** panel, select **New**, then **Repository location…**.

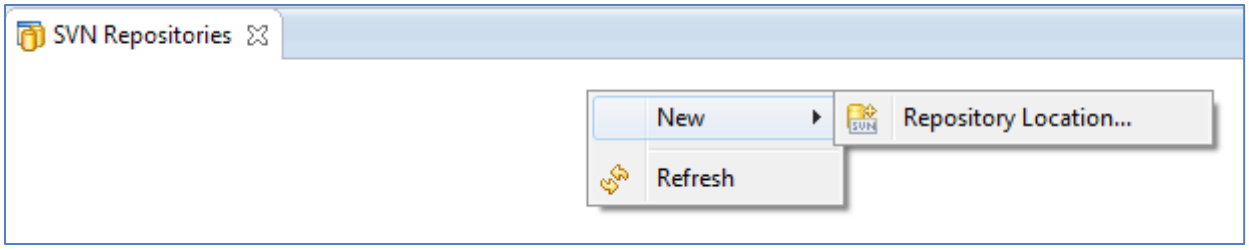

Figure 8 - Adding new repository location

    2. Insert https://whibo.googlecode.com/svn/trunk/ in **URL** text box.

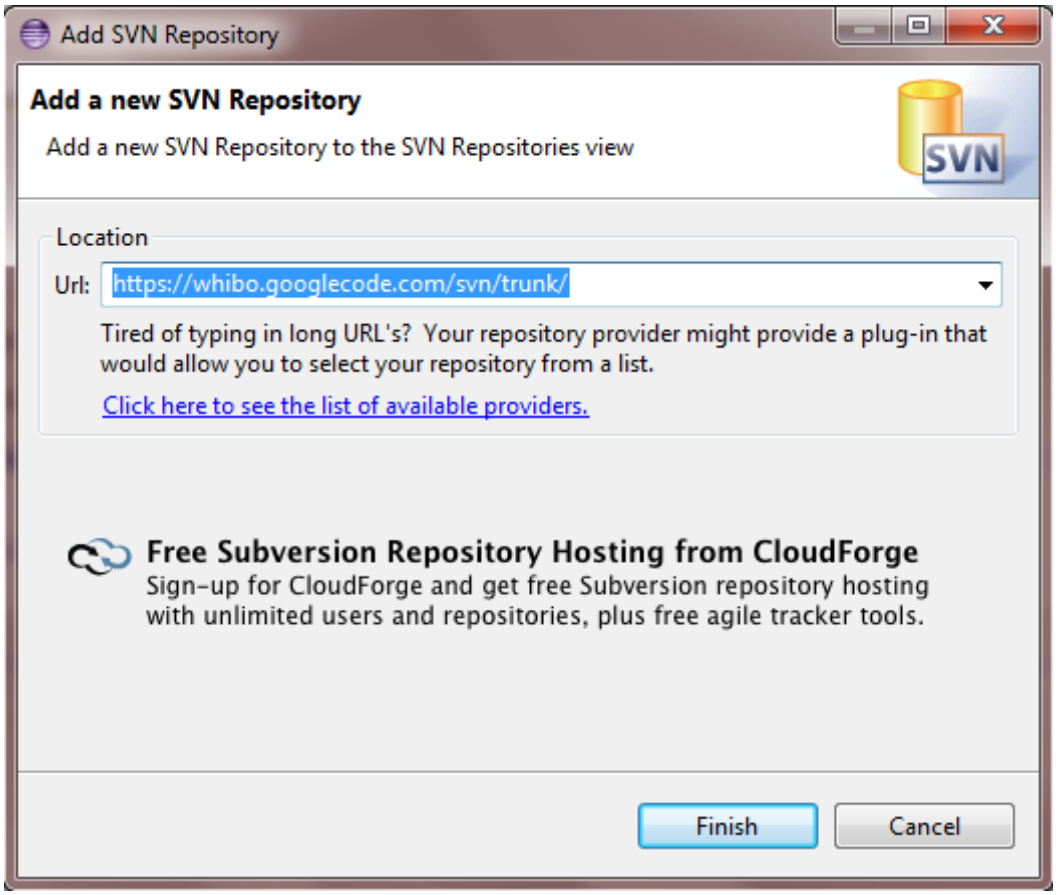

Figure 9 - Adding WhiBo repository location

    3. Click **Finish** button.

    4. Right click on **WhiBo** repository in **SVN Repositories** panel.

    5. Select the **Checkout…** option.

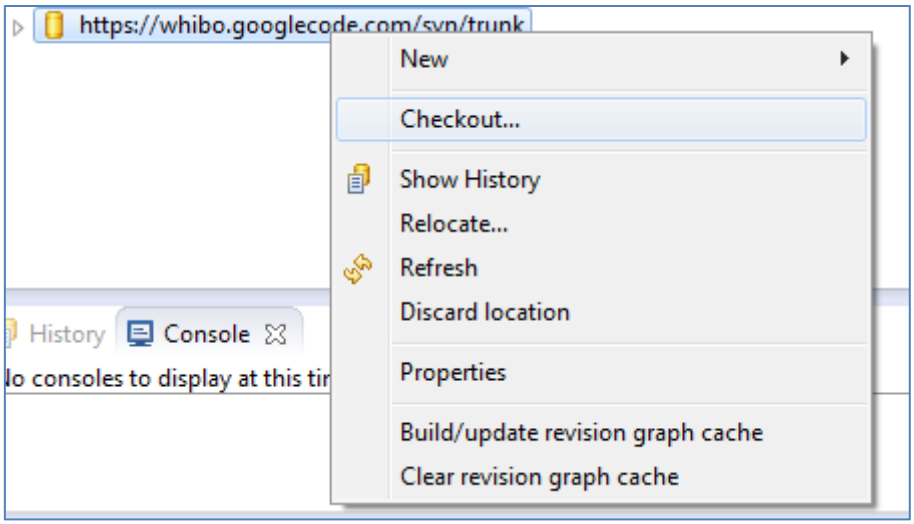

Figure 10 – Checkout of WhiBo project (1)

6. Select the **Check out as a project in the workspace** option and enter a project name.

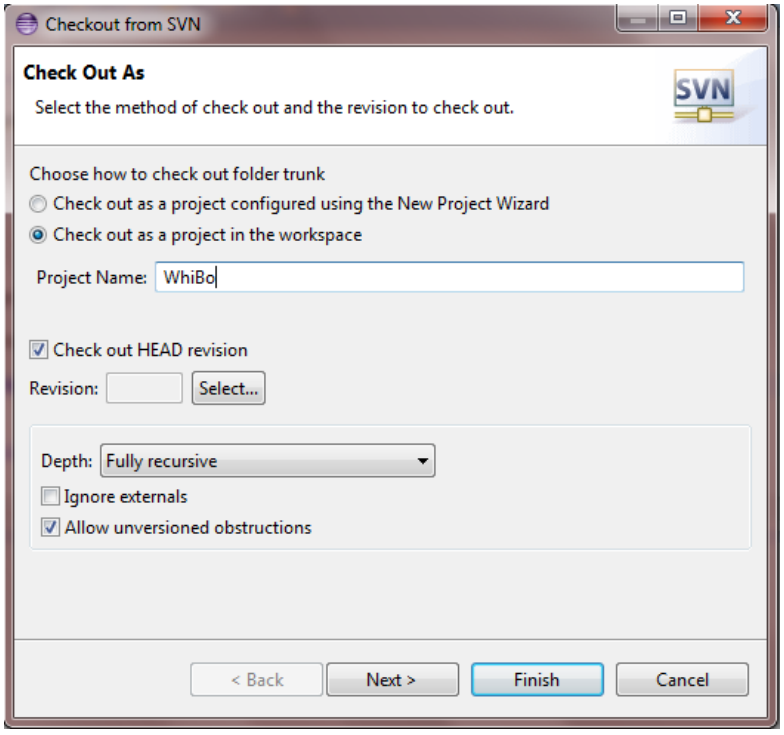

Figure 11 – Checkout of WhiBo project (2)

7. Select workspace where you wish to save project.

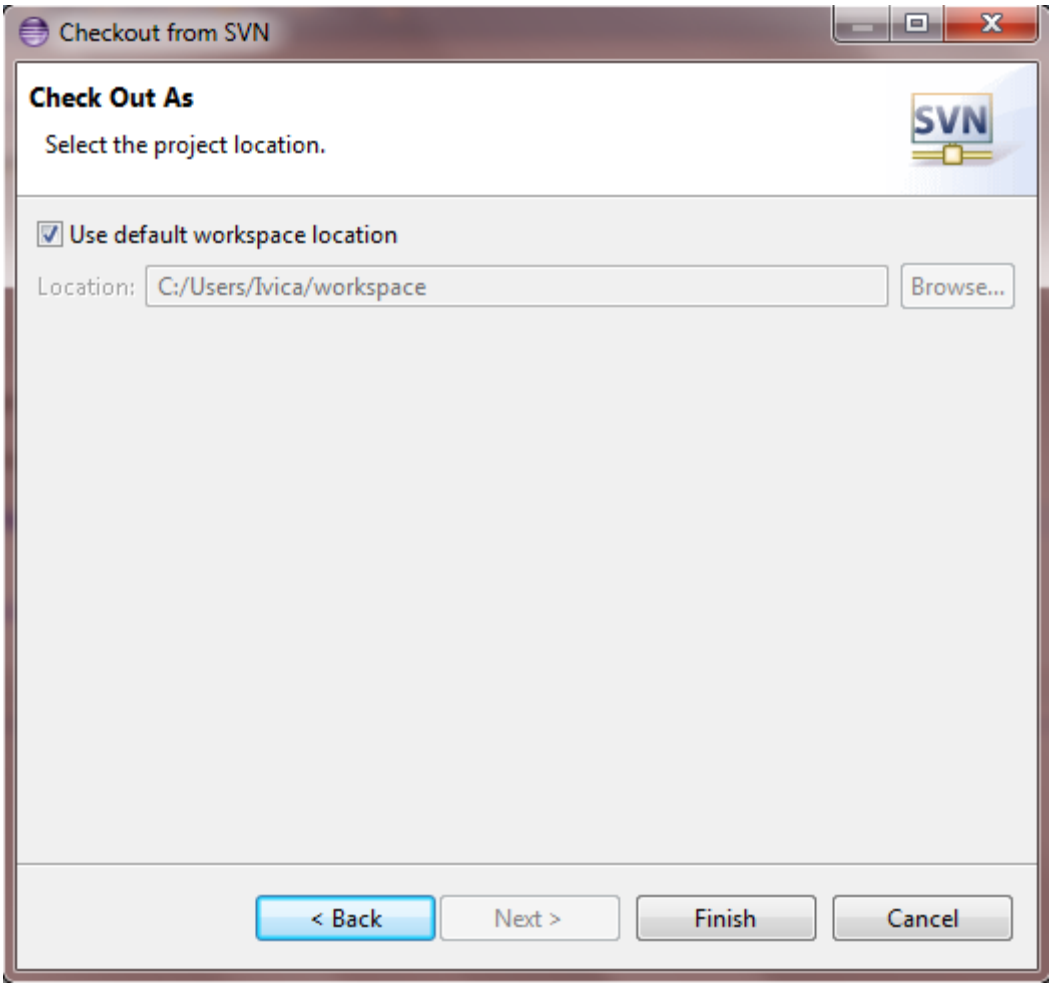

Figure 12 - Selecting workspace location

8. Click **Finish button.**

9. **WhiBo project will show up in Package Explorer panel.**

4. Similarly, **RapidMiner** project needs to be imported as project. URL for **RapidMiner** project is http://svn.code.sf.net/p/rapidminer/code. Currently, **RapidMiner** version is called **Unuk**.

5. After importing **RapidMiner** project it needs to be referenced in **WhiBo** project.

    1. Right click on **WhiBo** project.

    2. Click **Properties**.

    3. Select **Java Build Path** on left side and then **Project** tab on central panel.

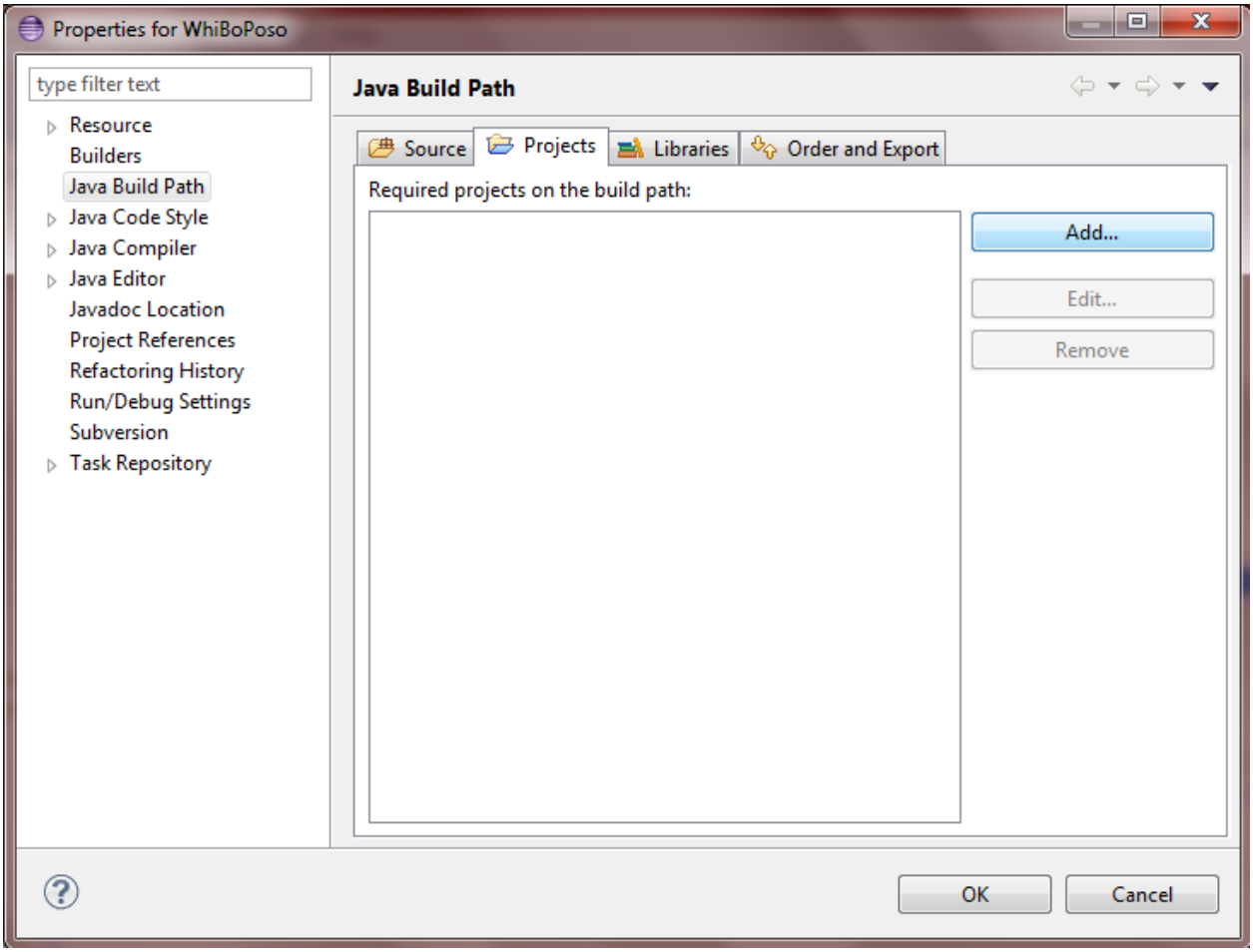

Figure 13 - Importing RapidMiner project into WhiBo project

4. Click **Add…** button.

5. Select proper **RapidMiner** version.

6. Click **OK** button on **Project Selection** panel.

7. Click **OK** button on **Properties** panel.

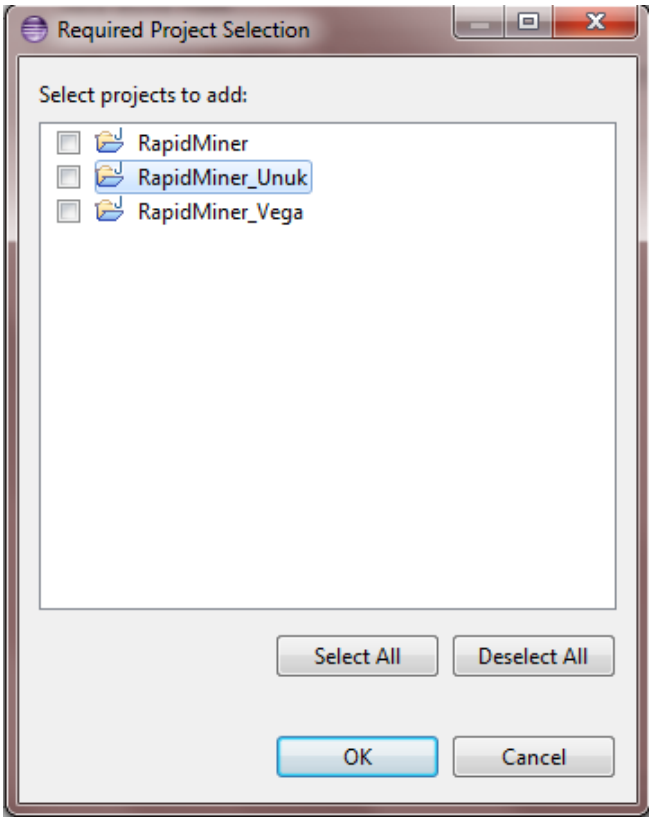

Figure 14 - Selecting RapidMiner version

6. Open **build.xml** file of **WhiBo** project.

7. Make sure that fifth line contains proper **RapidMiner** project (in this case it should be:
`<property name="rm.dir" location="../RapidMiner_Unuk" />`)

8. Right click on **build.xml** file and select **Run as…->Ant Build**. With this step **WhiBo** extension is building in **RapidMiner** project, so it can be used in that project.

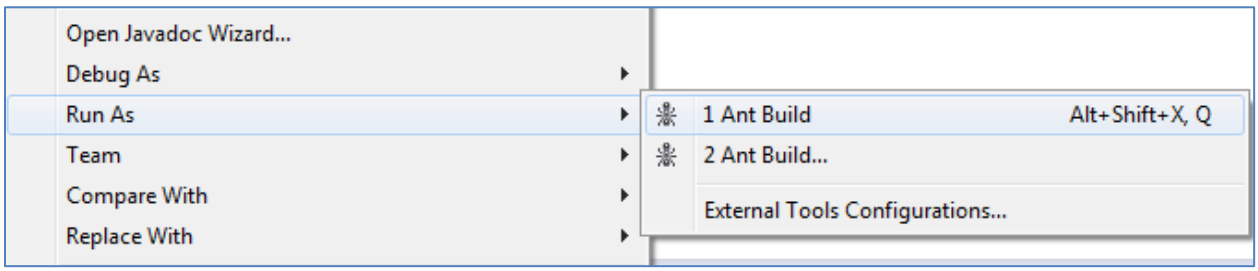

Figure 15 - Building WhiBo project

9. Right click on **WhiBo** project and select **Run as…->Java Application**.

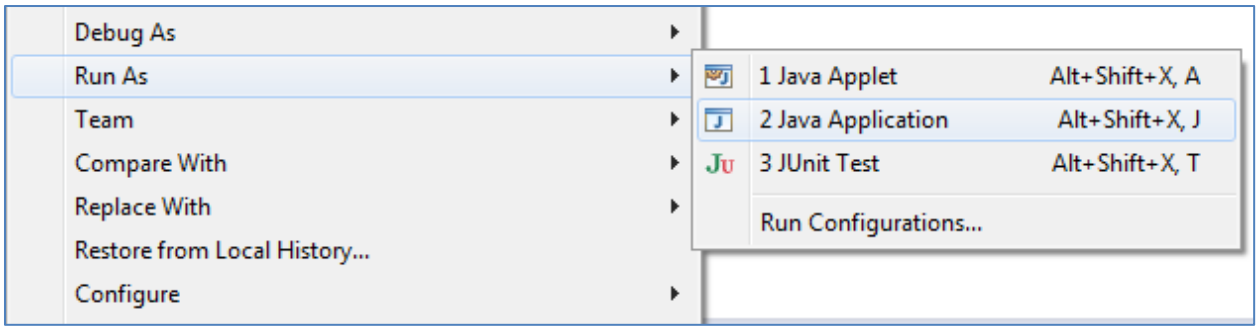

Figure 16 - Running WhiBo project

10. Select **RapidMinerGUI** class.

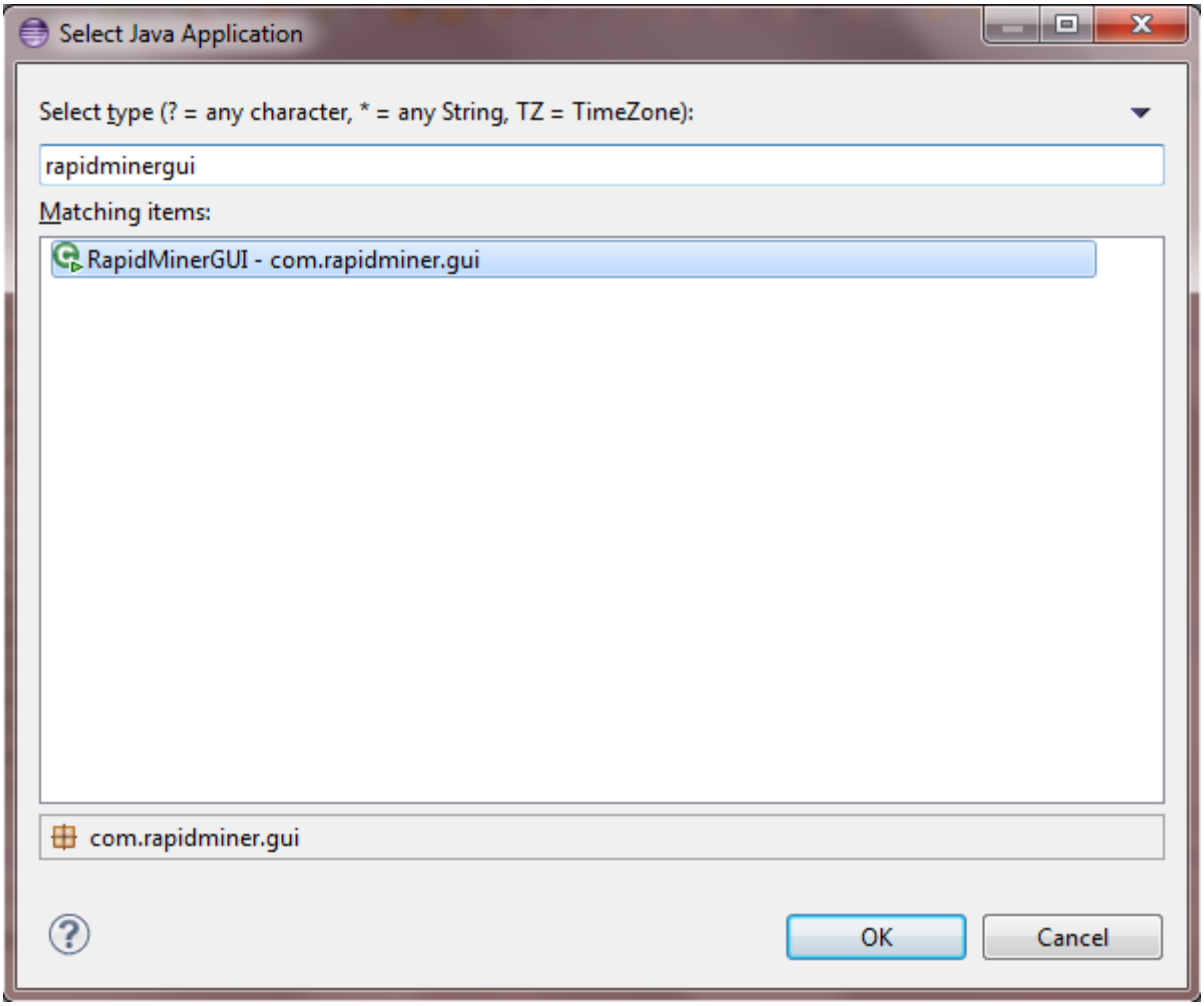

Figure 17 - Main class of WhiBo project

11. **RapidMiner** will start and **WhiBo** can be used.

For any information about configuration and extending WhiBo project you can contact us on e-mails (which can be found on the website) or on forum (which is also on the website).