Kalibro (Version 1.0)
User Guide

Fabio Kon – IME / USP (fabio.kon@ime.usp.br)
Paulo Meirelles – IME/USP
Carlos Morais – IME/USP
Vinícius Daros – IME/USP

10/20/2010

This work is partially funded by EU under the grant of IST-FP6-034763 – QualiPSo Project

www.qualipso.org

Table of Contents

1 INTRODUCTION

Source code metrics have been proposed since 70s, but they have not been fully used in software development. One reason is that most metrics do not have known thresholds. Metric tools show isolated numeric values, which are not easy to understand. Also, the interpretation of these values depends on the context.

Kalibro aims to improve the use of source code metrics. It is designed for easy integration with a metric collector tool and show the results in a friendlier way. For that, it allows a metric specialist to create a configuration of thresholds associated with qualitative evaluation, including comments and recommendations. These configurations are used to enhance metric results interpretation.

Kalibro is delivered with the proper configuration to run using Analizo (softwarelivre.org/mezuro/analizo) - a free and multi-language toolkit - as the source code analysis tool. Analizo supports the extraction and calculation of a fair number of source code metrics, generation of dependency graphs, and software evolution analysis. It efficiently parses source code written in C, C++ and Java.

In order to have an easy-to-use tool which provided information for multi-language source code, Analizo was designed to support the use of external tools as extractors. Doxyparse is the main extractor and grants great performance. It is based on Doxygen, a widely used open source tool for the generation of documentation through the code comments made in many programming languages (complete list on doxygen.org).

Kalibro Desktop provides Kalibro features with a graphical user interface. Usually, projects and configurations are stored on a database, and it is required to have the metric collector tool installed locally. Altough, Kalibro features may also be provided as a Web Service and Kalibro Desktop can be used as a graphical client of this service.

Kalibro Service is the service which provides Kalibro features. Through extractors, it can be connected to Spago4Q (spago4q.org) – a free platform to measure, analyse and monitor quality of products, processes and services.

2 USING THE SOFTWARE

At first Kalibro run, the blank main screen is shown. From this screen, it is possible to access all features.

Figure 1: Kalibro main screen

2.1 Configuration

Figure 2: Configuration menu

From Configuration menu, it is possible to create a new configuration, open or remove one created previously. A configuration is a set of thresholds associated with qualitative evaluation, including comments and recommendations. These configurations are used to enhance metric results interpretation.

By clicking on *New* option, a new window will pop where a name to the new configuration must be given.

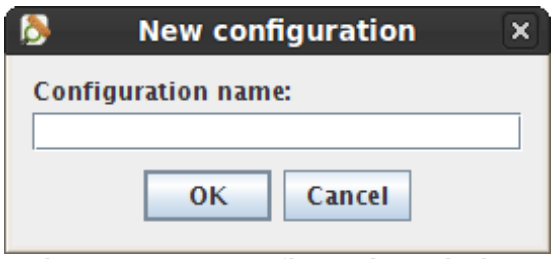


Figure 3: New configuration window

Once it is done, a new window with configuration characteristics will be displayed.

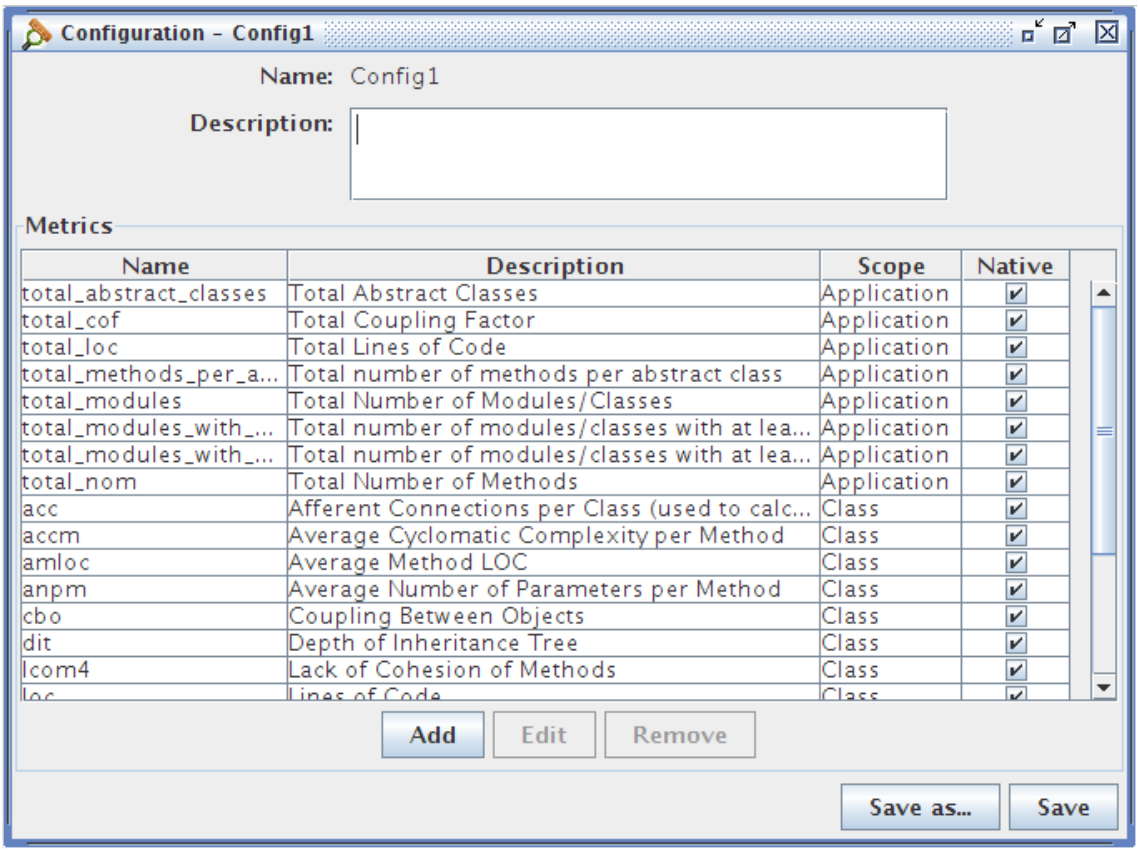


Figure 4: Configuration description window

In this window it is possible to set a description to the configuration and to verify the metric list. The metric list shows all metrics associated with the configuration, their description, the scope (or granularity, e.g. Class or Method) and a flag indicating if the metric is native from metric extractor, Analizo by default, or not.

Besides that, from this window it is possible to manage configuration's metrics by three basic operations: *Add*, *Edit* and *Remove*.

- **Add**: By default, all native Analizo metrics are loaded when a new metric is created. But, if a native metric were previously removed or not loaded,

when *Add* button is clicked, a window will pop where the kind of metric must be select. Otherwise, compound type will automatically be selected.

If the chosen metric extractor does not supply metrics which meet your specific needs, you are able to create new metrics based on native ones.

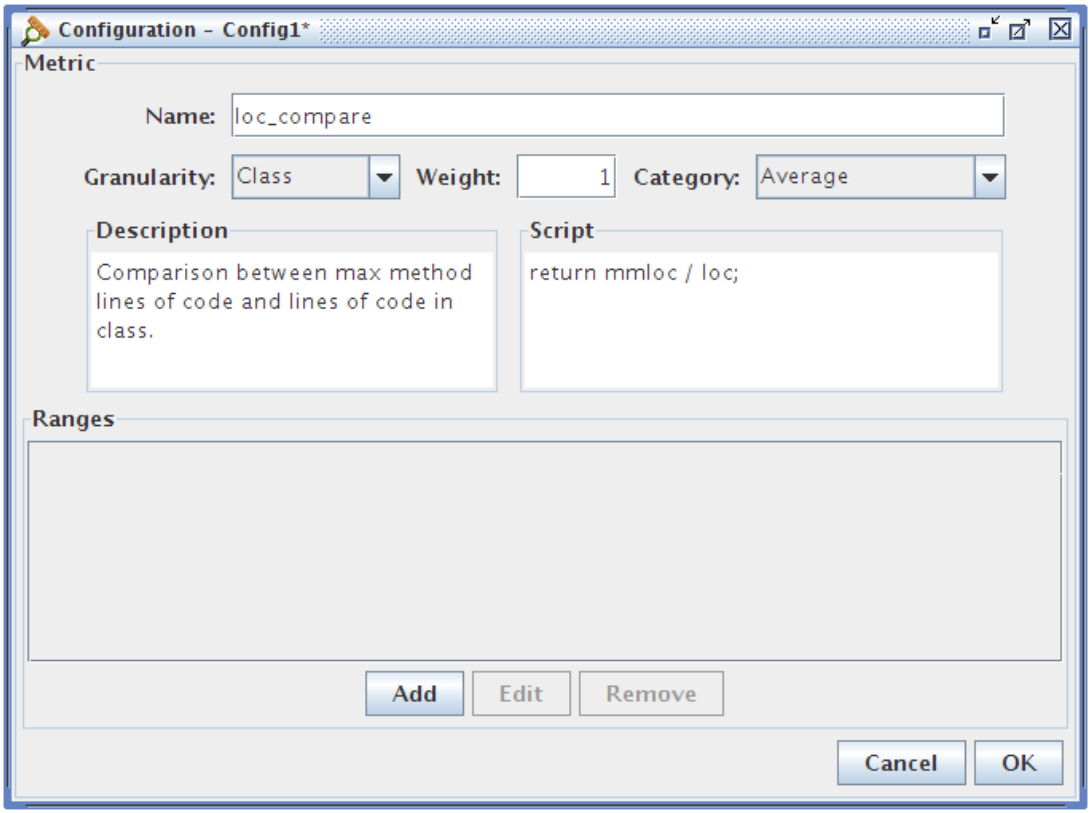


Figure 5: New metric example

Besides *Name* and *Description*, in this window it is possible to set metric *Granularity*, a *Weight* this metric will have when calculating result grade, the *Category* it best fits and, finally, the *Script*.

The script used to calculate the metric must be written in Java Script syntax and it is possible to get any other metric result just by using the metric name as variable. Figure 5 shows a simple example of compound metric.

Below the *Description* and *Script* boxes, there is the *Ranges* area. The ranges are value intervals used to make metric results friendlier giving them an understandable meaning.

By clicking on *Add* button a window as shown on Figure 6 will appear.

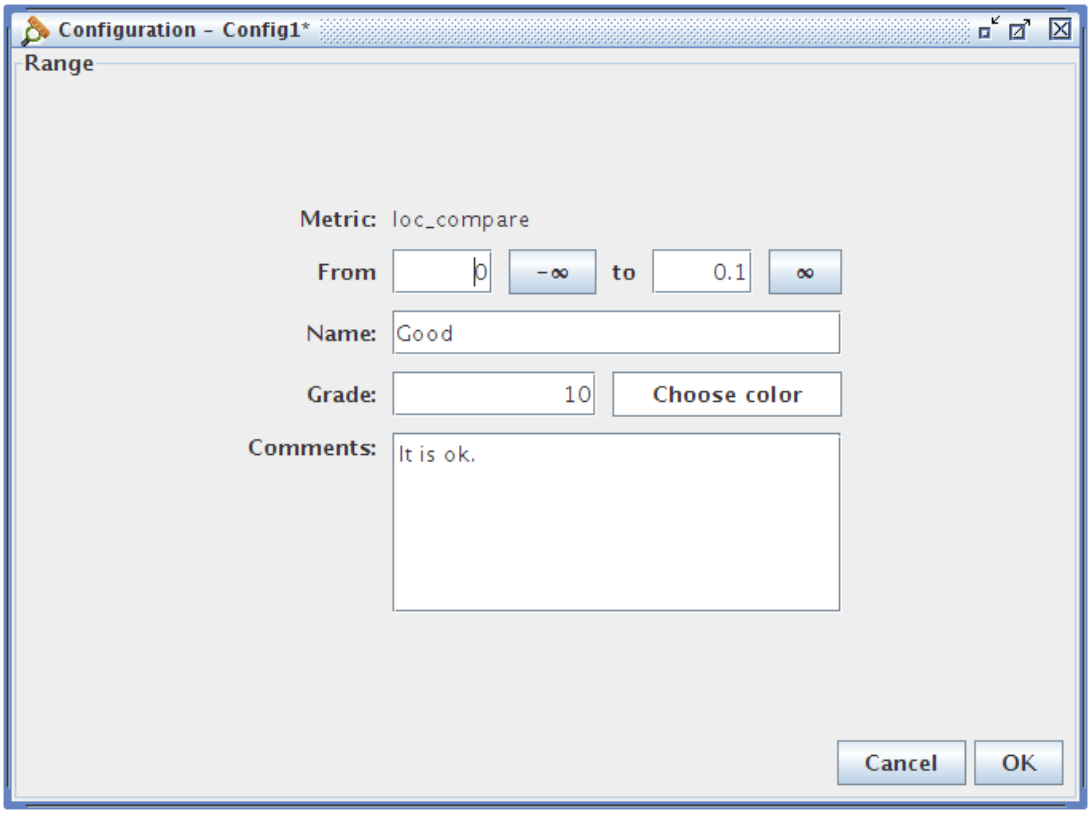


Figure 6: New range

In this window, it is possible to set the boundaries of the range, a *Name*, a *Grade*, a *Color* and write a *Comment* which will be displayed when a result fits in that range.

IMPORTANT: the numeric interval in ranges are left-closed and right-opened. For example, in mathematical notation, the interval on Figure 6 is [0, 0.1[

Once some ranges are set, the metric screen will be like Figure 7:

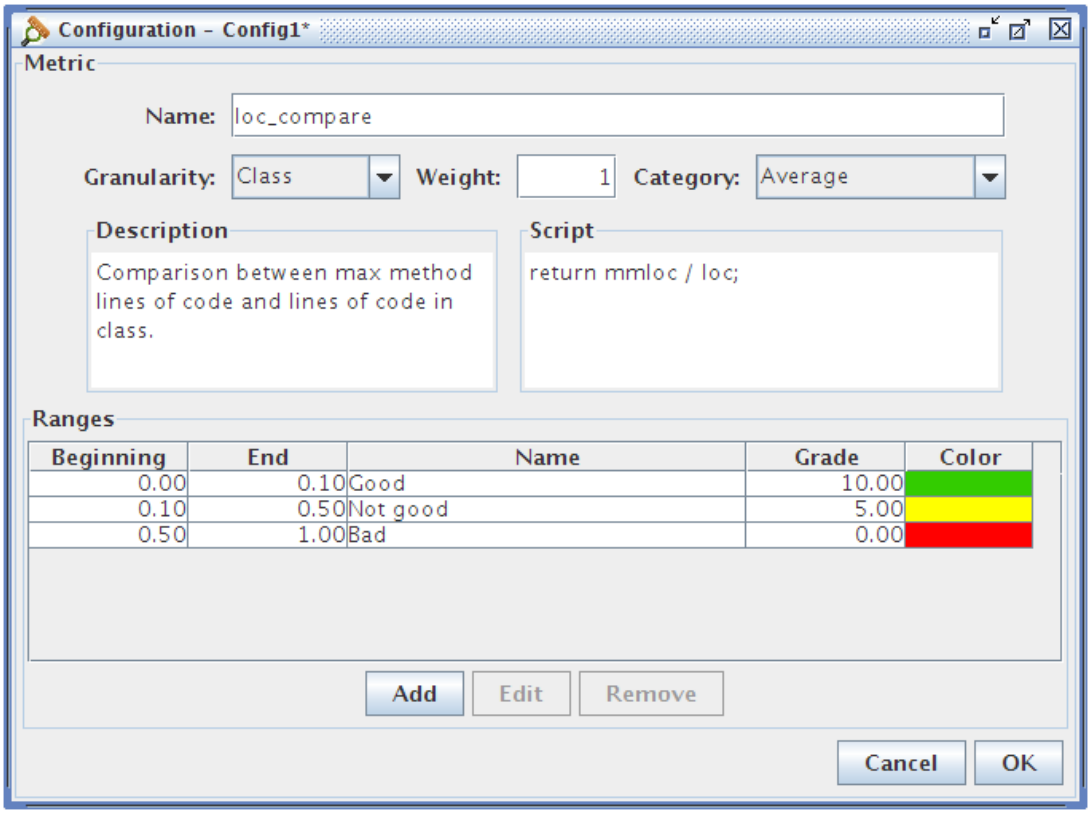


Figure 7: Metric with ranges

- **Edit**: By clicking on edit button, the same window shown on Figure 5 will pop. But all fields will only be editable if the chosen metric is compound. Otherwise, *Name, Granularity* and *Script* will be locked.

- **Remove:** It is possible to remove any metric, compound or native.

2.2 Project

Figure 8: Project menu

In *Project* menu, you can create a new project, open or remove one. Opening and removing a project are basically selecting the name on a list. So only *New* project option will be explained in detail.

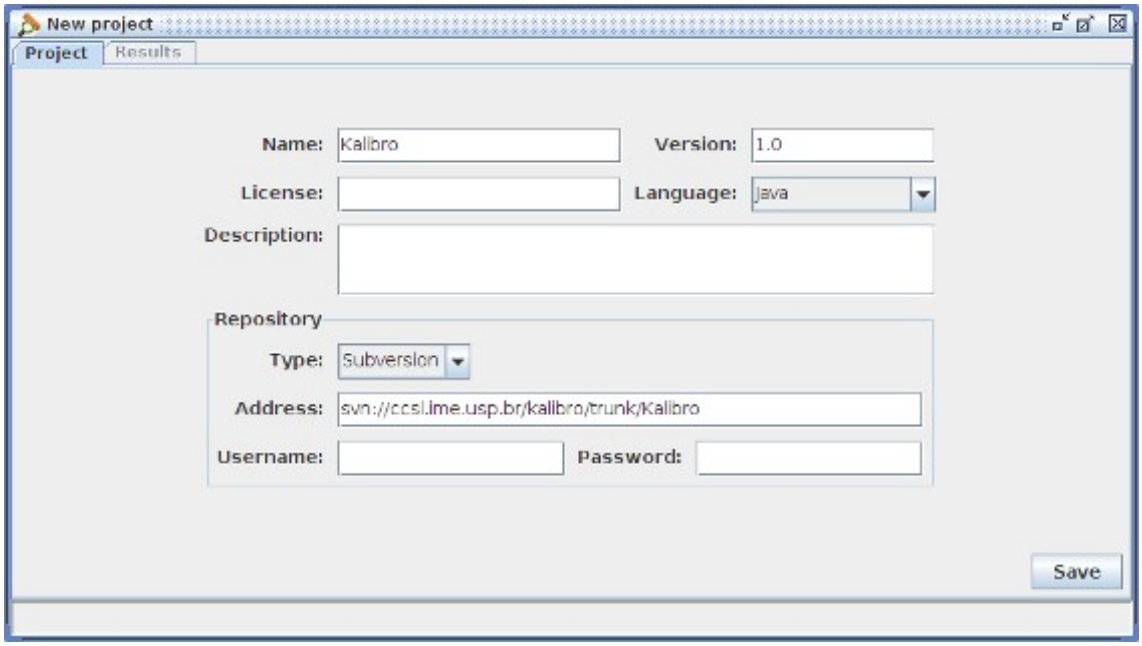


Figure 9: New project window

Once *New* option in *Project* menu is clicked, the window shown on Figure 9 will pop. There, *Name*, *Version* and *Language* fields must be filled.

In *Repository* section, the path to source code must be specified. If the code is stored in a local folder, then select *Directory* as *Type* and fill *Address* field with the full path to the code. When the code is stored in a subversion repository, change *Type* to *Subversion*, fill *Address* with repository URL and, if necessary, set *Username* and *Password* used to checkout the project. Besides these options, it is also possible to give an *Address* pointing to an URL of a

.tar.gz or .zip file. In that case, *Type* must be selected as *Tarball* or *Zip*, respectively.

After that, the project will be loaded and metrics will be calculated. When this task is finished, the window switches automatically to *Results* tab as shown bellow.

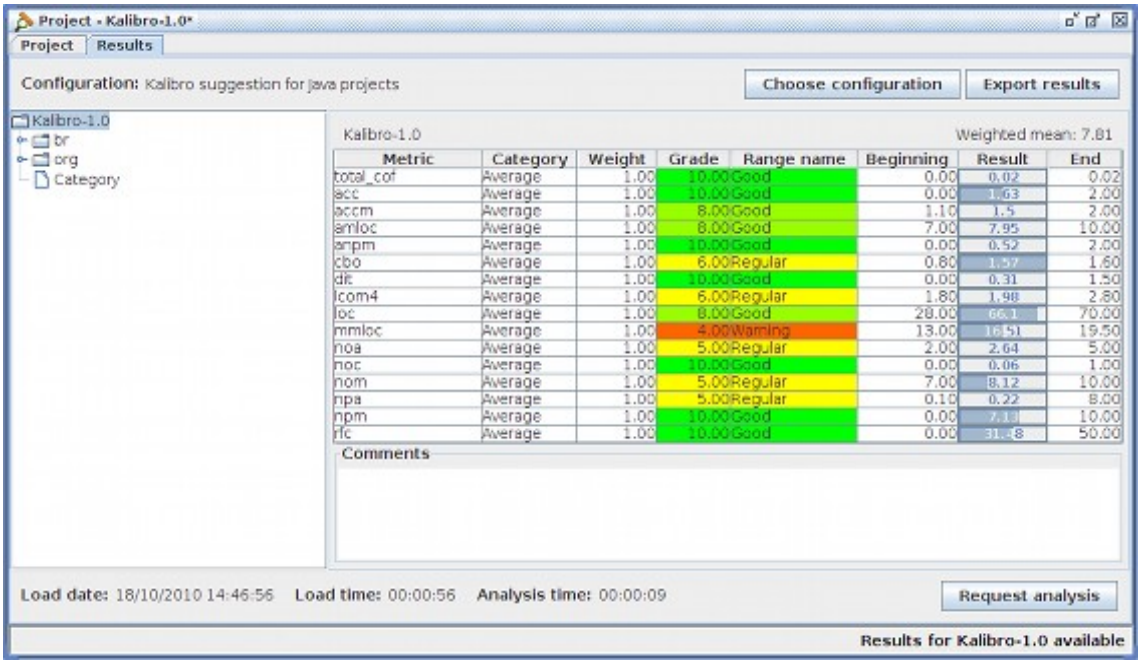


Figure 10: Results tab

On the left, there is a file tree where it is possible to browse through packages and classes. When the entire application or a package is selected, the result table will display statistical values for each metric, such as average and median. But, if a class is selected, its specific results are shown.

By clicking on a metric result in table, the *Comments* box will show the comment associated with the range in which the metric result fits. So, if the range has any hint about what could be done to improve that result, it will be easy to see the message. Figure 11 shows an example.

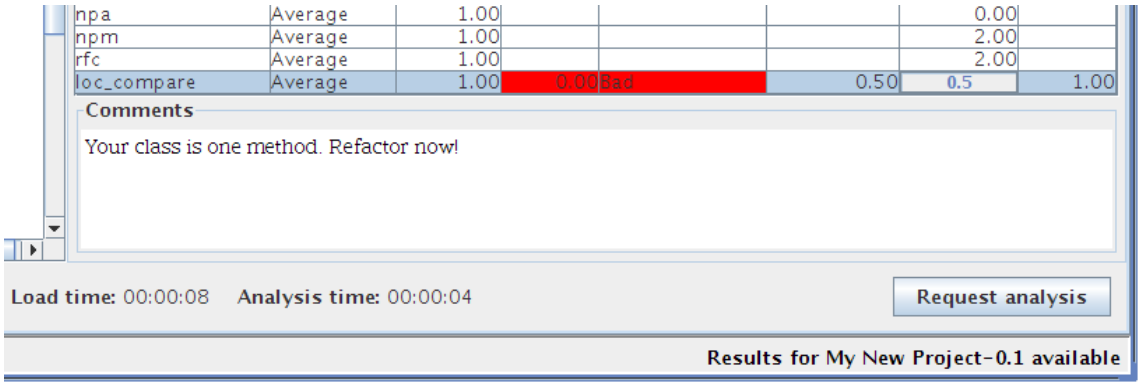


Figure 11: Zoom at comment about a bad result.

When the project has changed, all metrics must be recalculated. It is possible to do so just by clicking on *Request analysis* button.

If you have more than one configuration, it is possible to choose which one will be used clicking on *Choose configuration* button. Notice that there is no need to rerun the metrics calculation once the results will not change. What is going to change is only the ranges used to compare these results.

Other Kalibro feature is that when a configuration is chosen, this association with the project is stored and, next time that same project is open, the chosen configuration will be loaded automatically.

Finally, by clicking on *Export results* button, it is possible to save a csv file with the results shown in the result table.

2.3 Settings

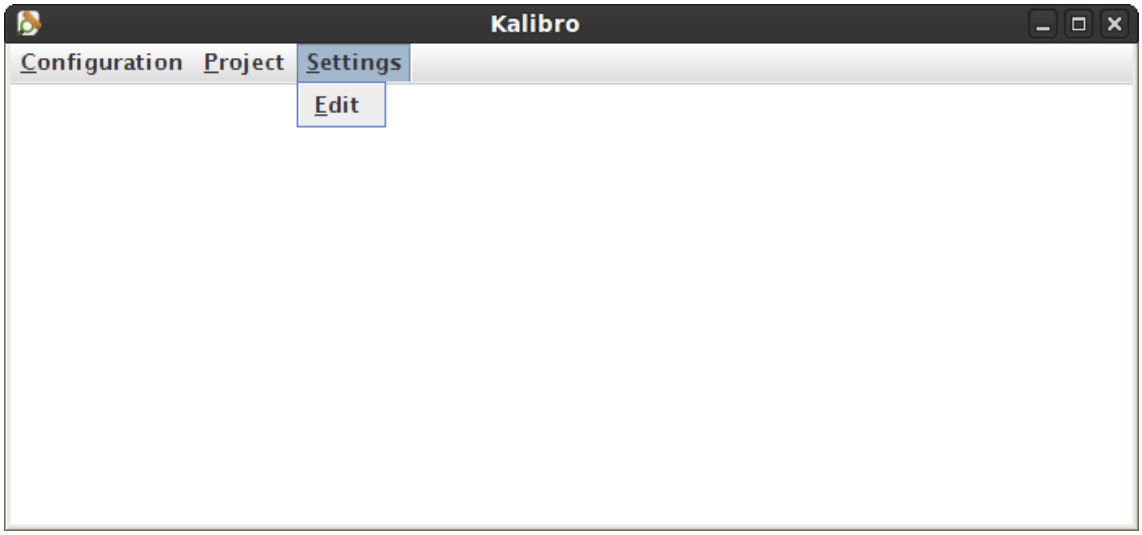


Figure 12: Settings menu

Before trying to change Kalibro settings, all configuration and project windows must be closed. Once it is done, *Edit* menu is unlocked.

By clicking on *Edit* menu, the window shown on Figure 13 will pop and there some Kalibro settings may be changed.

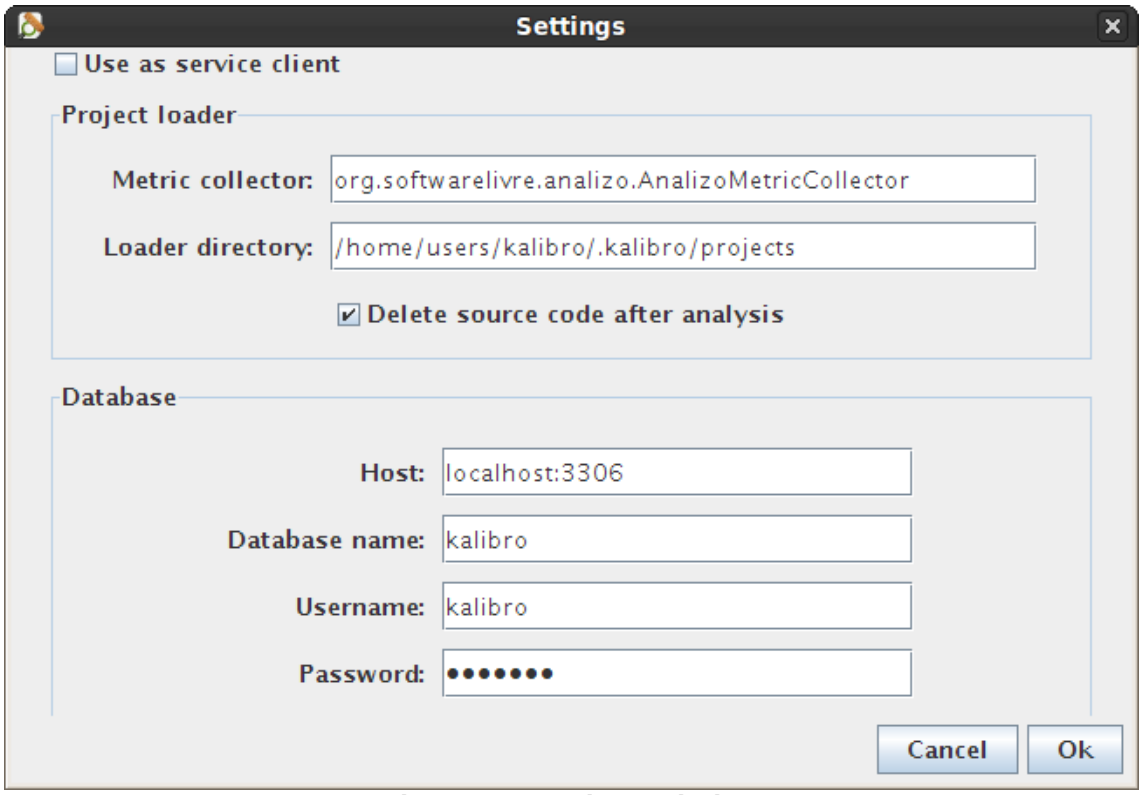
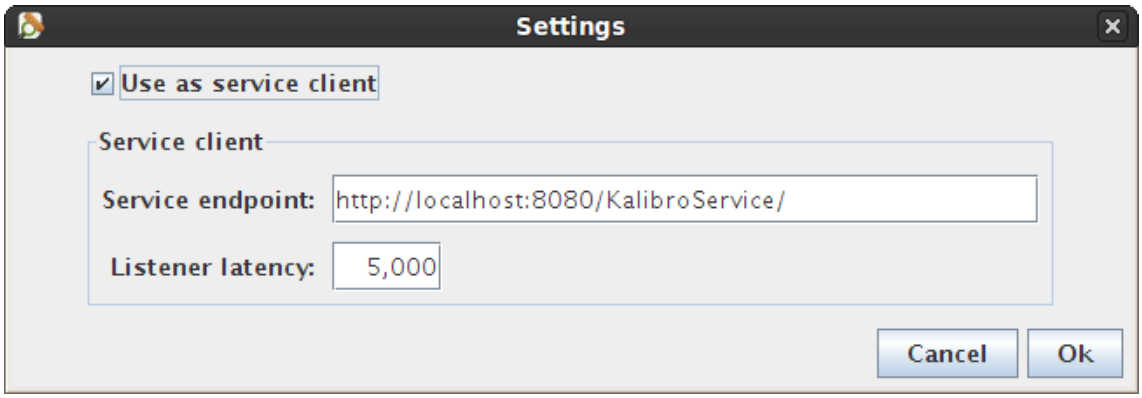


Figure 13: Settings window

The first option available is *Use as service client* check box. If it is checked, the window will show different options.



In service client mode, Kalibro will try to connect to a server which will run the analysis and only send back the results. The server address is set in *Service endpoint* field. The frequency that Kalibro will request results from server can be set in *Listener latency* field.

If Kalibro is not in service client mode, it is needed to configure how projects will be loaded and analysed and the database where results will be stored.

In *Project loader* box, *Metric collector* indicates the class used as metric extractor. *Loader directory* indicates the full path to a directory which will store

all source code loaded. If there is no need to keep the code after having the analysis result, *Delete source code after analysis* check box may be unchecked.

In *Database* box, it is possible to specify database host address in *Host*, the *Database name*, and authentication information in *Username* and *Password* when it is required.

3 TROUBLESHOOTING

Kalibro Service generates 2 log files at Tomcat logs folder (`kalibro_debug.log` and `kalibro_info.log`) corresponding to the debug and info level messages.

Every message show its date and time and the name of the thread from which it was generated. Every error ocurred detected by Kalibro classes are reported to the `kalibro_info.log` file.

References

[1] http://subversion.apache.org/

[2] http://softwarelivre.org/mezuro/analizo