Kalibro (Version 1.0)
Architecture Overview

Fabio Kon – IME / USP (fabio.kon@ime.usp.br)
Paulo Meirelles – IME/USP
Carlos Morais – 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 (www.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 www.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 (www.spago4q.org) – a free platform to measure, analyze and monitor quality of products, processes and services.

2 ARCHITECTURE OVERVIEW

2.1 Projects separation

Under Kalibro respository there are 9 Eclipse [3] projects as listed below:

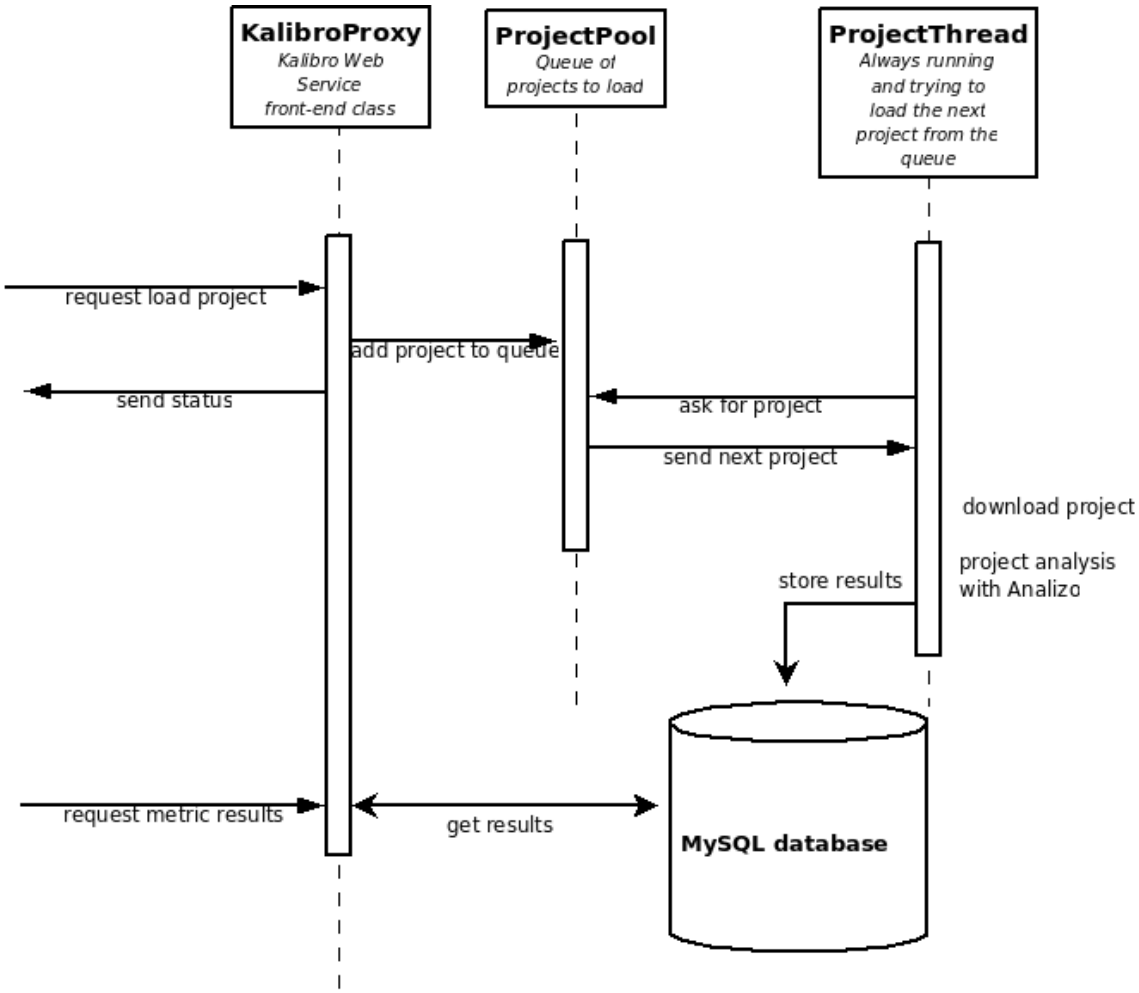
1. KalibroCore: the heart of Kalibro. Contains all the logic, database access and metric collector interaction.
2. KalibroDesktop: the swing interface. Contains basically components and controllers which communicate either directly to the core or to the service client.
3. KalibroService: the Kalibro Service.
4. KalibroServiceClient: here most classes are automatically generated. The only classes to be updated are in the `service.client` package. They manage the communication with Kalibro Service.
5. KalibroServiceSupport: used by both KalibroService and KalibroServiceClient. Contains the DTOs (Data Transfer Object).
6. KalibroSpago: contains the Kalibro Extractor for Spago.
7. KalibroTests: contains all Kalibro automated tests.
8. Libraries: contains the dependencies.
9. Resources: contains documents, development guides, scripts and other utilities.

As said, all internal logic remains on KalibroCore. `KalibroFacade` was designed to serve as the entry point to the system. Classes from outside KalibroCore should basically access only this Facade and the basic types at `kalibro.core.types`. There are 2 implementations of this interface: `Kalibro` answers the requests directly and `KalibroServiceClient` transfer requests to a remote Kalibro Service.

2.2 Basic workflow

When the analysis of a project is requested, it goes to the queue in the `ProjectPool`. On the first time the pool is used it starts the ProjectThread, which runs continuously taking projects from the queue, loading and analyzing them.

Every such step is observed by the `ProjectListeners` registered. Project listeners receives notification when the state of a project changes. The state indicates where in this whole process the project is (e.g. waiting on queue, loading, being analyzed).



2.3 Persistence

The package `persistence` contains classes for persisting and obtaining kalibro objects. The DAOs (Data Access Objectss) are the interface to access the database. The package `persistence.mysql` is the Mysql implementation. The tables were built based on the classes at `kalibro.core.types`. The command for creating all tables are in the `Database.sql` file. If the tables do not exist, these commands are run on the database by the first time the DAO factory is initialized.

2.4 Loaders

Loading a project is basically getting the source code from the repository address and putting it on the temporary project folder for analysis. `ProjectLoader` is the parent class for all loaders, and there is a loader for each type of repository supported by Kalibro. The projects are associated to its respective loader via reflection, so the loader's name must be on the form <repository_type>Loader.

2.5 Integration with Spago4Q platform

Refer to the installation guide for instructions on how to import Kalibro extractors and document into Spago4Q.

`KalibroSpagoExtractor` obtains results of the loaded projects and then loads the projects in Spago queue.

References

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

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

[3] http://www.eclipse.org/