Aplicação de Práticas de Usabilidade no processo de desenvolvimento empírico de software.

Jônatas Medeiros Rodrigo Medeiros Paulo Meirelles
LAPPIS – Universidade de Brasília, Brasil

Métodos Empíricos de Desenvolvimento de Software

- Software livre é uma filosofia que trata programas de computadores como fontes de conhecimento que devem ser compartilhados
- A utilização de métodos ágeis no desenvolvimento empírico tem como características intrínsecas a flexibilidade e rapidez nas respostas a mudanças

Testes Automatizados

A automação de testes é uma prática ágil, eficaz e de baixo custo para melhorar a qualidade do software. Foram utilizadas neste estudo práticas de TDD (*Test-Driven Develepment*) e BDD ((Behavior Driven Development). Segue os testes utilizados:

- **Testes de unidade:** testes de correção responsável pelos menores trechos de código com um comportamento [**?**];
- **Testes funcionais:** estes que tem como objetivo veri-ficar a eficiência dos componentes de um sistema [**?**];
- **Testes de aceitação:** testes para verificar se um módulose comporta como foi especificado [**?**];

Usabilidade

- **Design centrado no usuário:** filosofia baseada nas necessidades e interesses dos usuário, com ênfase em fazer produtos usáveis e inteligíveis.

Kalibro Architecture

- **KalibroCore** is the heart of Kalibro Metrics. It contains all logic, including database access, metric collector tool interaction, statistics computation and compound metric scripts evaluation. It includes a Java abstract class (KalibroFacade) to access all its functionality.
- **KalibroService** is a Web Service interface for using Kalibro.

Kalibro Plug-in for Mezuro networking

- A social network to track source code metrics.
  - This environment promotes an open and collaborative networking to analyze hundreds of thousands software projects, specially Free Software, through an automated tracking of their source code repositories.
- Mezuro is a powerful environment to enhance Kalibro features, using the social network potential.
  - The idea is based on the fact that people improve their writing skills when they read good books and papers. Similarly, software engineers can increase the quality of their source codes reviewing good and clean codes.
  - a developer can find through source code metrics software projects implemented in the same context and language, they can compare their source code characteristics and know other related projects.
- Users just need to give a source code repository URL.
- Users can access the source code analysis report from an asynchronous way, i.e. when they wish or need.
- The history of source code metric values and analysis are recorded.
- All free and public project analysis are available to any user.
- Any user can suggest metric threshold configurations and share them on the Mezuro network.

- Download source code from local and remote zip and tarball (.tar, .tar.gz, and tar.bz) files.
- Creation of configurations, i.e., a set of metrics for being used in the evaluation of a software source code.
- Creation of ranges associated with a metric and a qualitative evaluation.
- Creation of new metrics (via JavaScript) based on the ones provided by the metric collector tool.
- Calculation of statistics results for higher granularity modules (e.g. average LOC of classes inside a package).
- Possibility of exporting results to a CSV (comma-separated values) file.
- Calculation of a grade for the source code analysis projects, based on given weights for each metric and grades for ranges. This allows cross-project comparisons.
- Possibility of making interpretation more user-friendly by associating colors with ranges.

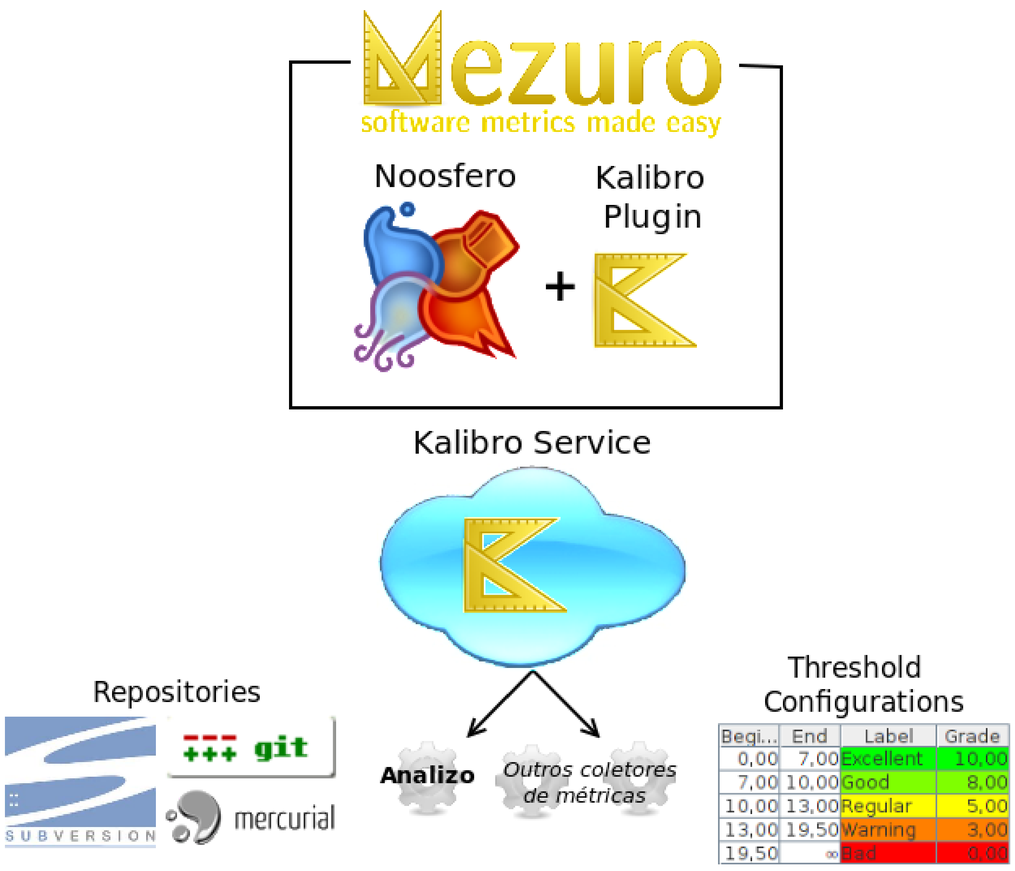

Figure: kalibro interation diagram.