IDE
Software Engineering Project Description

Alice Rixte, Kévin Le Run

September 2016

1 General description

This project aims to design an IDE (Integrated Development Environment) for OCaml and possibly other languages. The goal is to organize the code in a different way than the conventional file structure. The functions and classes should be arranged in an intuitive way so as to facilitate software design. However the creation and design of the text editor is not within the scope of this project.

Browser The browser allows navigation through classes and functions in the same way than file browsers. With a function selected, the user can edit its implementation, its documentation, its tags, or its test functions and add categories within modules. The user should also be able to interactively run the function or its tests in isolation.

Function/Method finder Allows finding functions by name, by documentation keywords and by tags. Also allows finding functions by giving an example of the expected inputs and outputs (through the help of a simple language). As an example, giving 3 and 4 as inputs and asking for a function returning 7 could yield a function computing the addition of integers.

Dependency graph Allows the user to see the code structure through a graph of class and module dependency.

Import/Export Allows the user to switch between this IDE and a conventional project structure. It analyses the `.ml` files to extract the code structure.

Language Plugins Allows adding more language support to the IDE with extensions that can be implemented separately from the main program, so that plugins can be produced by the community. It should be able to change the terminology of code structure (modules-¿packages, methods-¿functions) and the hierarchy while still preserving the file-system-like browsing and search features. It should redefine import and export features to accomodate to the new language.

2 Detailed objectives

- α Graphical User Interface (GUI)
- α Saving, loading, exiting
- α Browsing the structure
- α Export
- α Basic research
- α External text editor integration
- β Import
- β Language plugins (with at least a Java plugin)
- β Advanced research
- β Interactive function execution
- β Compiler integration
- β Interactive testing system (run tests only on selected modules and functions)
- ?? Dependency graph
- ?? Debugger integration
- ?? Documentation generation

3 Comments

```
// This is a comment
```