Project: Plagiarism-detector Specifications

Olivier Duménil
Antoine Jacquin-Ravot

Table of Contents

I/ Scope of the Project

This project aim to produce an application able to help teachers detect plagiarism in programming assignments. It should provide the following features: detection methods modularity, ease of access, results readability.

The creation of new detections algorithms is out of scope. Our main objective is to produce a program able to present the results of multiple detection algorithms in a meaningful way.

II/ Technical specifications

Here are described the needs on which we are going to focus on for the development of this project.

- Possibility to apply multiple detection algorithms on a same data set.
- Easy to read and interpret results for the user.
- Have the results of multiple detection algorithms displayed at once.
- Possibility to add new detection algorithm to the application in an easy way.
- Normalised way to parameterise different detection algorithms.
- Provide efficient plagiarism detection.

III/ Proposed design specifications

This is a proposed design specification for the project knowing the problems we want to address.

A/ Core application

a. Menu Bar
   - i. File
     1. Exit: exit the program
     2. Export: open the "Export" window
   - ii. Edit
     1. Cut
     2. Copy
     3. Past
     4. Undo
     5. Redo
   - iii. Modules
     1. Manage modules: open the "Manage Modules" window
     2. Select module(s): open the "Select Modules" window
   - iv. Sources
     1. Add sources folder(s): open a directory browser
     2. Add sources file(s): open a file browser
     3. Manage sources: open the "Manage Sources" window
     4. Select sources skeleton: open the "Select Sources Skeleton" window
   - v. Analysis
     1. Start/Resume: start/resume the analysis
     2. Stop: cancel the analysis
     3. Pause: pause the analysis
   - vi. Help
     1. About: open the "About" window

        2. Documentation: open the “Documentation” window

b. Configuration view
    i. Module Selector
        1. Name of each module
        2. Checkbox to activate/deactivate the module
        3. Module configuration button: open the configuration form provided by the module in the “Module Configuration” window
    ii. Sources Browser
        1. Sources drag’n’drop area
        2. Add sources file(s) button: open a file browser
        3. Add sources folder(s) button: open a directory browser
        4. Delete button: delete the selected file(s) and/or folder(s)
    iii. Analysis Status
        1. Start/resume button
        2. Stop button
        3. Pause button
        4. Progress bar
        5. Status label

c. Pop-up windows
    i. Export
        1. Export format picker
        2. Export button
        3. Cancel button
    ii. Manage Modules
        1. List of currently loaded modules
        2. Checkbox to activate/deactivate the module
        3. Module configuration button: open the configuration form provided by the module in the “Module Configuration” window
        4. Add module button: open a file browser and load the selected file(s)
        5. Delete module button: unload the selected module(s)
    iii. Manage Sources
        1. Sources drag’n’drop area
        2. Add sources file(s) button: open a file browser
        3. Add sources folder(s) button: open a directory browser
        4. Delete button: delete the selected file(s) and/or folder(s)
    iv. Select Sources Skeleton
        1. Sources Skeleton radio list
        2. Add sources skeleton button: open a file browser
        3. Delete sources skeleton button: delete the selected file(s)/folder(s)
    v. About
        1. Display information about the software, its version, the project
    vi. Documentation
        1. Display the documentation file
    vii. Module Configuration
        1. Display the configuration form provided by the module
        2. Validate button

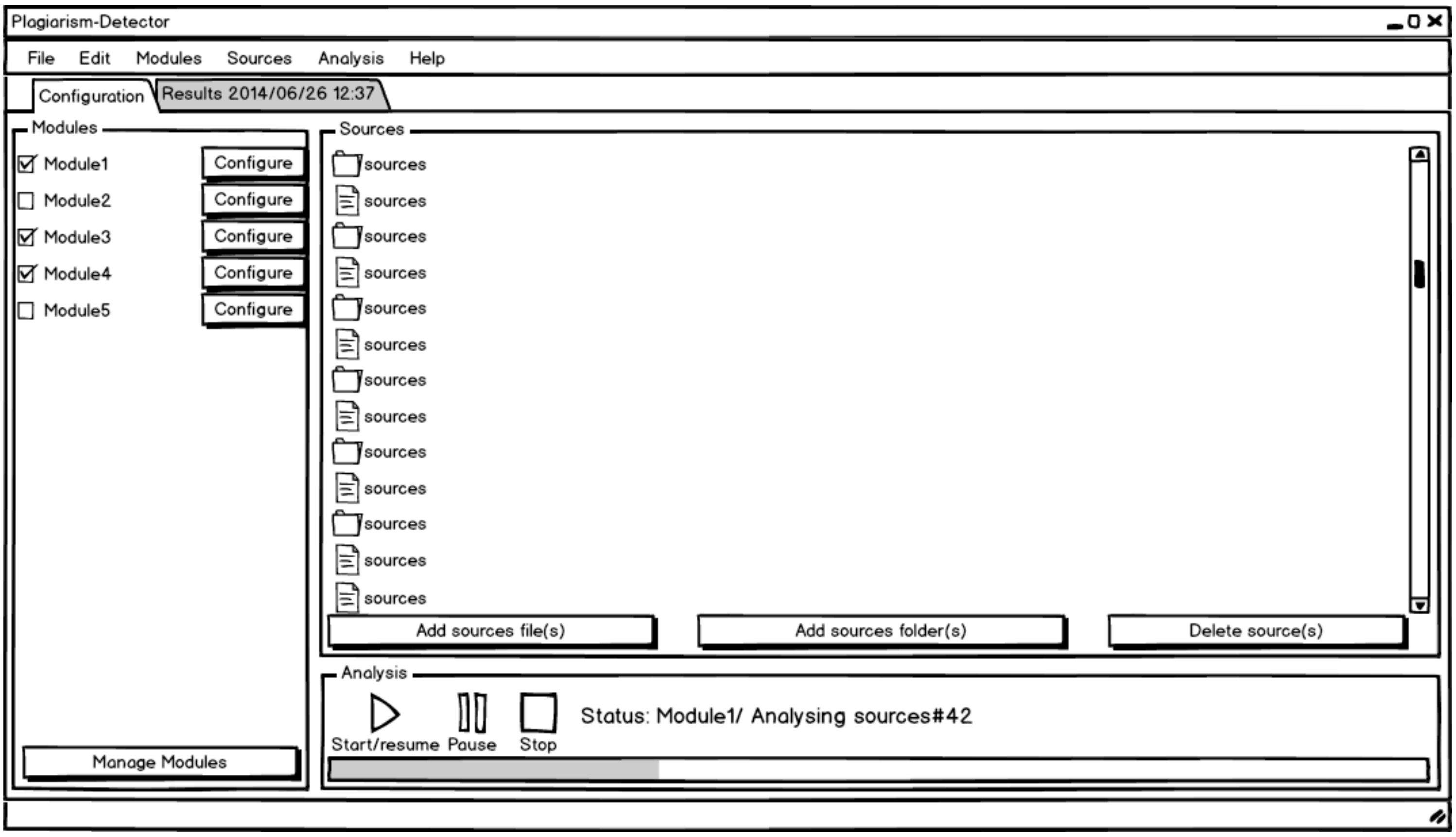
d. Result view
    i. Display the results

B/ Application Programming Interface

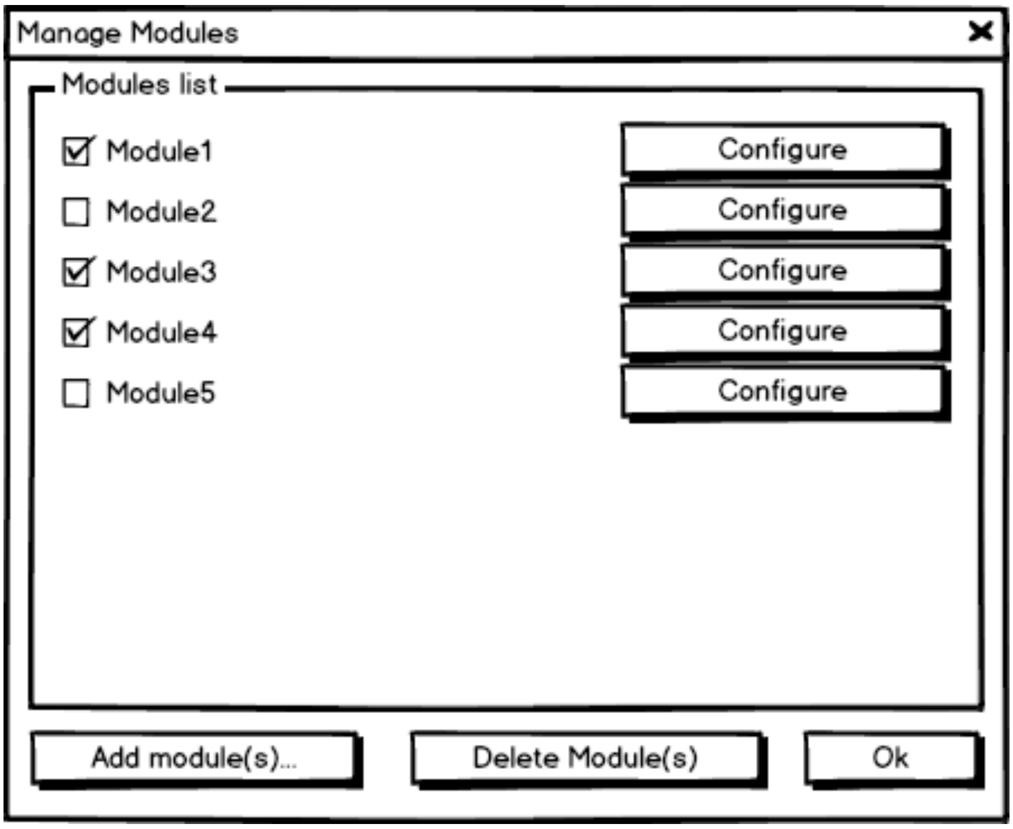
The detection modules must implement the following methods to be used by the core application

a. Module API
    i. GetModuleInformation: Send the name of the module, its version and miscellaneous information about the module
    ii. GetParameterForm: Send the form to be used to configure the module
    iii. SetParameters: set the parameters to be used by the module
    iv. Start: begin the analysis
    v. Resume: resume the analysis
    vi. Pause: pause the analysis
    vii. Stop: cancel the analysis
    viii. SetSources: set the folders and files to be analysed
    ix. GetResults: send the results of the analysis when availables
    x. SetDelegate: set the delegate to be used to notify the core application
b. Delegate API
    i. SetStatus: set the status of the analysis
    ii. SetProgress: set the progress of the analysis

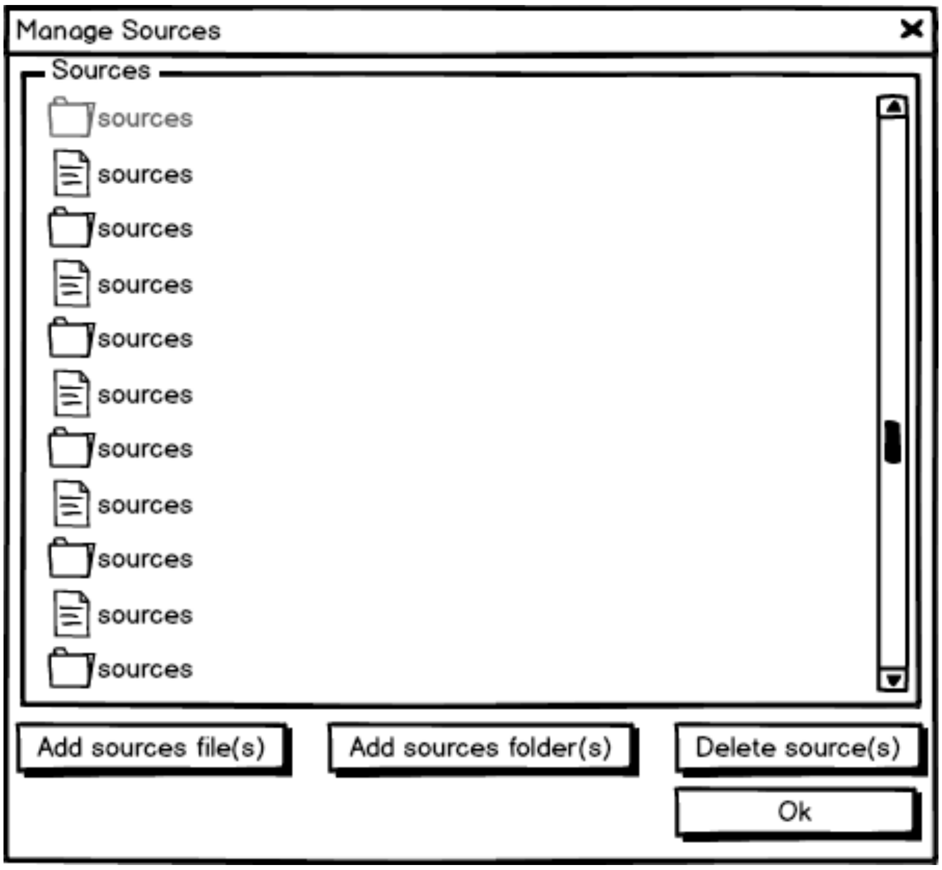
C/ Mockups



1/ Configuration view

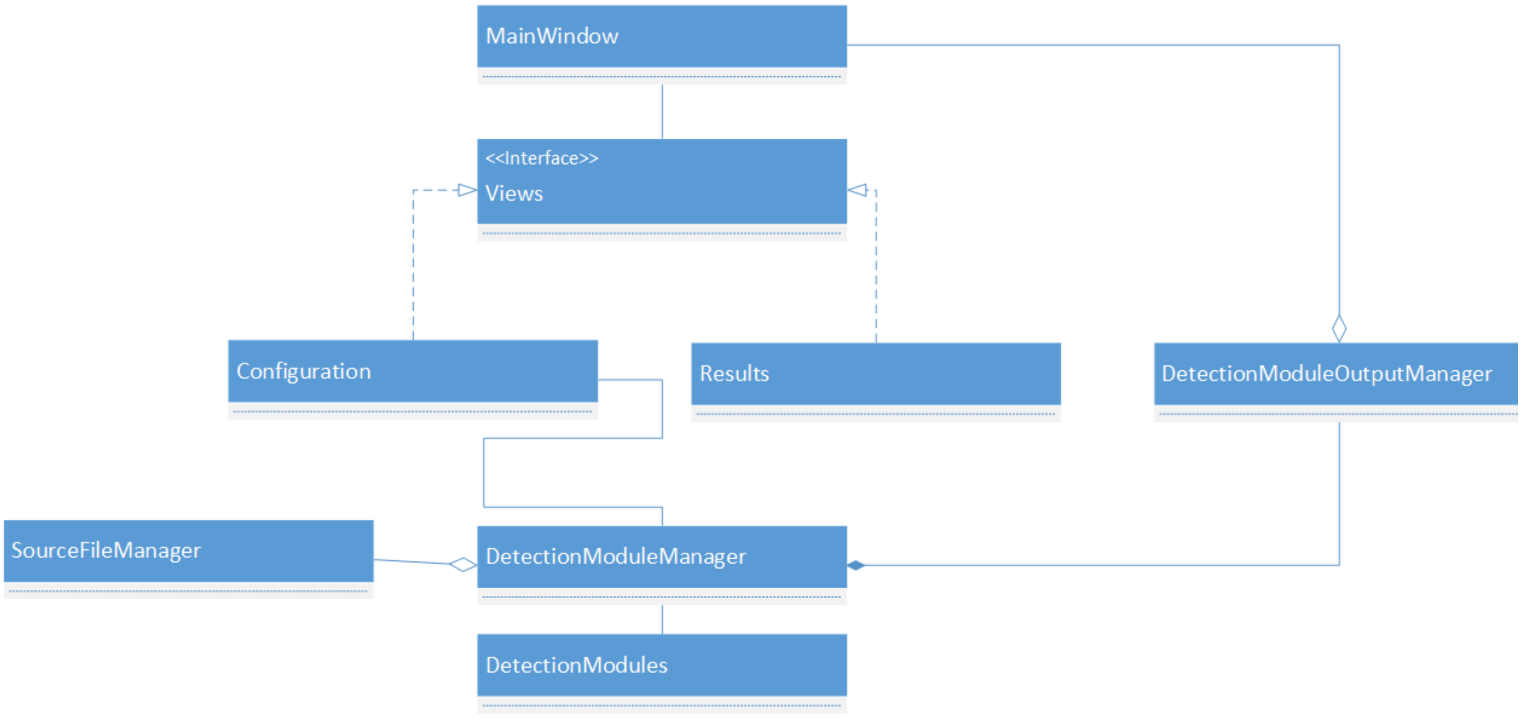


2/ Detection module manager



3/ Student project sources manager

IV/ Architecture of the project



The DetectionModuleManager will be responsible for almost everything in the application:

- Communicate with the configuration view: retrieving the user information from it, and giving back status updates from the DetectionModules.
- Retrieve objects from the SourceFileManager and passing them to the DetectionModules.
- Retrieve results from the DetectionModules and pass them to the DetectionModuleOutputManager to be processed.

The DetectionModuleOutputManager will ask the main window to create a view to render the processed results.

The MainWindow is responsible for the event handling and the instanciation of new views.

The Configuration view will be responsible for the configuration of the modules and the selection of source files to test.