Code Inspection Report

Tailor-made optimization problems and decision support Software Development Project

BSc in Computer Engineering
Academic Year 2017/2018 - 2º Semester
Software Engineering II

Group ES2-2018-EIC2-01
Nº 72702, Ana Pestana, EIC1
Nº 72897, Rodrigo Agostinho, EIC1
Nº 73174, Guilherme Azevedo, EIC2
Nº 73289, Susana Gomes, EIC1

ISCTE-IUL, Instituto Universitário de Lisboa
1649-026 Lisbon
Portugal

March 10, 2018

Table of Contents

Introduction

In this software project we intend to develop a platform for solving optimization problems and decision support.

The platform should allow the user to describe, through the graphical user interface (GUI), the characteristics of the problem to be solved.

The platform should, based on the characterization of the problem provided by the user, use the algorithms suggested by the user through the GUI, or automatically select the most adequate algorithms, to solve the problem. It should also compare the performance / quality of the solutions generated by these algorithms and display in the GUI the name of the algorithm that presents the best quality for solving the problem.

In addition, the platform should compare in a textual / numerical way and also graphically the best solution (s) found by the algorithm selected with the best known solution (s) provided by the user.

Code inspection – Graphical User Interface

This module focused on developing the storage and retrieval functionality of a Problem object from a XML file and the graphical user interface (GUI), which allows the users to characterize the optimization problem that they want to submit for evaluation.

Meeting date: *Meeting duration:* *Moderator:* *Producer:* *Inspector:* *Recorder:*	*10/03/2018* *180 minutes* *Susana Gomes* *Ana Pestana* *Guilherme Azevedo* *Rodrigo Agostinho*
Component name (Package/Class/Method):	*Packages frames, objects and utils*
Component was compiled:	*Yes*
Component was executed:	*Yes*
Component was tested without errors:	*Yes*
Testing coverage achieved:	*Yes*

Code inspection checklist

(See "*Sprint1_JavaInspectionChecklist*" document)

Found defects

Found defect Id	Package, Class	Description
1	frames.DecisionVariablesPage	When selecting the option “Import from file” a JOptionPane informing the user of the constraints for the chosen document pops-up. If the user closes this message instead of clicking “ok” it should return to the Decision Variables Page instead of proceeding to open the JFileChooser.
2	frames.DecisionVariablesPage	We will have to include the possibility of establishing domain conditions in order to comply with the specification (project statement).
3	frames.FitnessFunctionPage	When Uploading a Jar File we should restrict the search to files with JAR extension.Also, for consistency’s sake, we should configure JFileChooser to open on the user’s Desktop.
4	frames.FitnessFunctionPage	We cannot allow the user to proceed to the next page until all optimization criteria are associated with a single fitness function.
5	frames.FitnessFunctionPage	If, after having selected some checkboxes, we return to the previous page, and again advance to this page, some checkboxes became erroneously disabled.
6	frames.KnownSolutionsPage	Due to the padding of the cells containing the name of the decision variable, the initial part of the content is occluded.
7	frames.SendEmailPage	After submitting the email we should return to the Home Center.
8	frames.SaveProblemPage	The file name is not a required field. By default, it takes the value of "ProblemName_Year.Month.Day_Hour.Minute.Second.xml".

In addition to the errors pointed out, other modifications will be necessary so that we can integrate the graphical user interface with other functionalities that we will develop in the next sprints.

Corrective measures

We will include a GUI fix task in the Sprint Backlog of the next iteration.

Conclusions of the inspection process

It is our team’s quality assessment that the component inspected will need minor corrections for the purpose of delivery and also that it will need major changes for integration with features to be developed in the next iterations.

It should be noted that the coverage of the unit tests is higher than the established minimum of 75% in all coverage metrics. The classes of the package "frames" could not be the target of JUnit tests, since they are classes of modeling of graphical interfaces.