Experiment – Analysis of Code Smell Agglomerations

Environment Configuration

In this experiment we will analyze a set of agglomerations of the SportsTracker application (http://www.saring.de/sportstracker/). The version to be analyzed will be 5.7, which can be downloaded from:

https://sourceforge.net/projects/sportstracker/files/SportsTracker/SportsTracker%205.7.0/

To compile it, we will need (at least):

- Java SE Development Kit (JDK) 7
- Eclipse Juno (>4.0)
- Maven 3.0.3
- Groovy-Eclipse 2.7.0 (https://github.com/groovy/groovy-eclipse)
    - Groovy-Eclipse Feature
    - Groovy Compiler 2.0 Feature
    - Groovy-Eclipse M2E integration

Tasks

1. For each of the following 5 agglomerations:
    a. Analyze the related source code (i.e., the different code smells that are part of the agglomeration in question and the affected Java classes)
    b. Identify the problems that you consider are associated with the code smells of the agglomeration (e.g. tangling, scattering, code duplication, high coupling, low cohesion, concentration of functionality, cycles, little abstraction, etc.). For each problem, provide a brief explanation of your logic or reasoning to recognize it as a problem. If possible, exemplify with specific cases of the code for such problems.
    c. Based on the previous set of problems, analyze the probability that the agglomeration "as a whole" can generate a critical or important maintenance problem for the system (either immediately or in the future). Express your opinion using a scale of 1 and 5 (1 = improbable, 5 = very likely). Justify your answer.

For the analysis, you can help yourself with the following table:

Agglomeration	**Related Code smells**	**Identified Design Problems**	**Global analysis of the agglomeration (Scale: 1-5)**
#1			
#2			
#3			
#4			

#5			

2. Once you have analyzed all the agglomerations, provide an order for them, from what you consider most critical for the system that you consider less critical.

Code Smell Agglomerations to Analyze:

- de.saring.sportstracker.gui.dialogs (Intra-component)
    a. ExerciseDialog.ok (intensive Coupling)
    b. FilterDialog.ok (intensive Coupling)
    c. OptionsDialog.ok (Feature Envy)
    d. OptionsDialog.setInitialValues (Feature Envy)
    e. OptionsDialog.setInitialValues (Intensive Coupling)
    f. OverviewDialog (Brain Class)
    g. OverviewDialog.addEquipmentTimeSeries (Dispersed Coupling)
    h. OverviewDialog.addExerciseTimeSeries (Brain Method)
    i. OverviewDialog.addExerciseTimeSeries (Dispersed Coupling)
    j. OverviewDialog.addSportSubTypeTimeSeries (Dispersed Coupling)
    k. OverviewDialog.createExerciseFilterForTimeStep (Feature Envy)
    l. OverviewDialog.mergeExerciseFilterIfEnabled (Feature Envy)
    m. OverviewDialog.mergeExerciseFilterIfEnabled (Intensive Coupling)
    n. OverviewDialog.updateDiagram (Brain Method)
    o. SportTypeDialog.editEquipment (Feature Envy)
    p. SportTypeDialog.editSportSubType (Feature Envy)
    q. SportTypeDialog.updateEquipmentList (Feature Envy)
    r. SportTypeDialog.updateSportSubtypeList (Feature Envy)
    s. SportTypeListDialog.deleteSportType (Feature Envy)
    t. StatisticDialog.changerFilter (Feature Envy)
    u. StatisticDialog.setFilterValues (Feature Envy)
    v. StatisticResultsDialog.setStatisticResults (Feature Envy)
- OptionsDialog (Intra-class)
    a. OptionsDialog.ok (Feature Envy)
    b. OptionsDialog.setInitialValues (Feature Envy)
    c. OptionsDialog.setInitialValues (Intensive Coupling)
- de.saring.sportstracker.gui.views.listview (Intra-component)
    a. BaseListView (Refuse Parent Bequest)
    b. BaseListView.createPopupMenu (Feature Envy)
    c. BaseListView.getTable (Shotgun Surgery)
    d. ExerciseListView (Refuse Parent Bequest)
    e. ExerciseListView.addColumnByIndex (Feature Envy)
    f. ExerciseListView.print (Feature Envy)
    g. ExerciseListView.updateView (Feature Envy)
    h. NoteListView.print (Feature Envy)
    i. WeoghtListView.print (Feature Envy)

- de.saring.sportstracker.data (Intra component)
    a. Equipment.getName (Shotgun Surgery)
    b. Equipment.setName (Shotgun Surgery)
    c. Exercise.getAvgSpeed (Shotgun Surgery)
    d. Exercise.getDistance (Shotgun Surgery)
    e. Exercise.getDuration (Shotgun Surgery)
    f. Exercise.getEquipment (Shotgun Surgery)
    g. Exercise.getSportSubType (Shotgun Surgery)
    h. Exercise.getSportType (Shotgun Surgery)
    i. ExerciseList.getExercisesForFilter (Feature Envy)
    j. ExerciseList.getExercisesForFilter (Intensive Coupling)
    k. ExerciseList.updateSportTypes (Feature Envy)
    l. ExerciseList.updateSportTypes (Intensive Coupling)
    m. SportSubType.getName (Shotgun Surgery)
    n. SportSubType.setName (Shotgun Surgery)
    o. SportType.getColor (Shotgun Surgery)
    p. SportType.getEquipmentList (Shotgun Surgery)
    q. SportType.getName (Shotgun Surgery)
    r. SportType.getSportSubTypeList (Shotgun Surgery)
    s. SportType.isRecordDistance (Shotgun Surgery)
    t. SportType.setName (Shotgun Surgery)
    u. Weight.getValue (Shotgun Surgery)
- Exercise (Intra-class)
    a. Exercise.getAvgSpeed (Shotgun Surgery)
    b. Exercise.getDistance (Shotgun Surgery)
    c. Exercise.getDuration (Shotgun Surgery)
    d. Exercise.getEquipment (Shotgun Surgery)
    e. Exercise.getSportSubType (Shotgun Surgery)
    f. Exercise.getSportType (Shotgun Surgery)

3. Based on the analysis done previously, make a ranking of the agglomerations being the first of them the one that you consider most critical for the maintenance of the system.