Synthesis of Code Anomalies – Eclipse Plugin

Synthesis of code anomalies is the systematic **search** and **summarization** of **information** about **code anomaly agglomerations**. Code-anomaly agglomerations are **groups** of **inter-relatedcode anomalies** that may **indicate** the presence of **design problems**. Code anomalies may be agglomerated (i.e. grouped) based on different forms of relationships. After detecting the occurrence of an agglomeration, the synthesis technique summarizes the information about this agglomeration. The summary includes: (i)the list of code anomalies that comprise an agglomeration, (ii) information about surrounding code elements, i.e. external elements somehow related to the agglomeration, but which are not an explicit member of it, and (iii) information about the agglomeration in past versions of the system.

Here we present a tool for the systematic synthesis of code anomalies. This tool was designed to help developers in diagnosing design problems in Java systems. Moreover, it is implemented as a plug-in for the Eclipse platform.

In this tool we implemented a number of features that collaborate with each other to satisfy the requirements for a synthesis technique. The types of Code Anomaly Agglomerationsupported by the tool are presented in Table 1. Next, we present an overview of the features that will be used in this experiment.

Table 1. Types of Code Anomaly Agglomeration

Name of the Agglomeration Type	Description
Intra-method	A method affected by two or more code anomalies; i.e. the code anomalies (taking part of the agglomeration) are located within a single method body.
Intra-class	A class (including its methods) affected by two or more code anomalies; i.e. the code anomalies (taking part of the agglomeration) are located within members of a single class.
Intra-component	A component that contains two or more classes affected by the same type of anomaly i.e. occurrences of the same type of code anomaly are located within classes of a single component.
Hierarchical	Two or more classes in a common inheritance tree (including interface implementation) that are affected by the same type of anomaly
Concern Overload	Classes that implement one or more cross-cutting concerns besides implementing the main concern of their enclosing component

Agglomerations View

The Agglomerations View provides all the features of the *synthesis technique* in order to support the diagnosis ofdesign problems. This is the main view that you will interact with

during this experiment. Figure 1 shows a screenshot of the Agglomerations View. As you can see, this view is separatedin two parts: the first part is called “Agglomerations”and it is shown on the left side; the second part is called Details and it is shown onthe right side. The Agglomerations partshows the agglomerations found in one or more projects according to their type. In Figure 1, for instance, this view is showing all the agglomerations found in the HealthWatcherOO_10_ExceptionHandling project. Once you click on the arrow in the project's node, all categories of agglomerations will be displayed (Concern-Overload, Hierarchical, Intra-Class, Intra-Component and Intra-Method in our example). Finally, when you click on one of the agglomeration categories,all the detected agglomerations of that category will be displayed. Each agglomeration will havea meaningful identifier. For example, there are three Intra-method agglomerations in Figure 2. The identifier of each intra-method agglomeration is the name of the affected method.

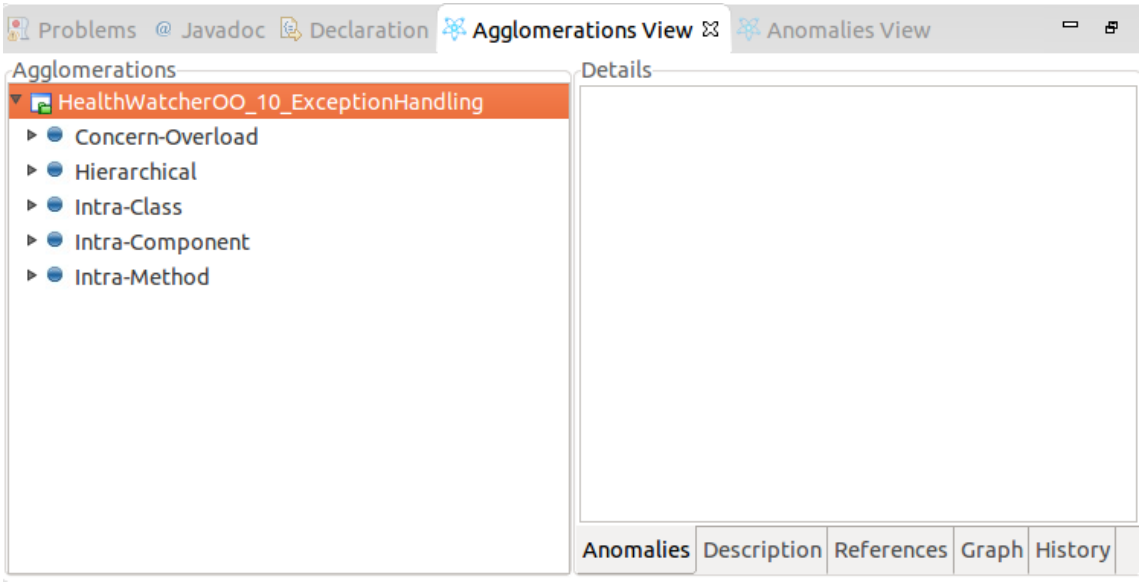

Figure 1. Agglomerations View

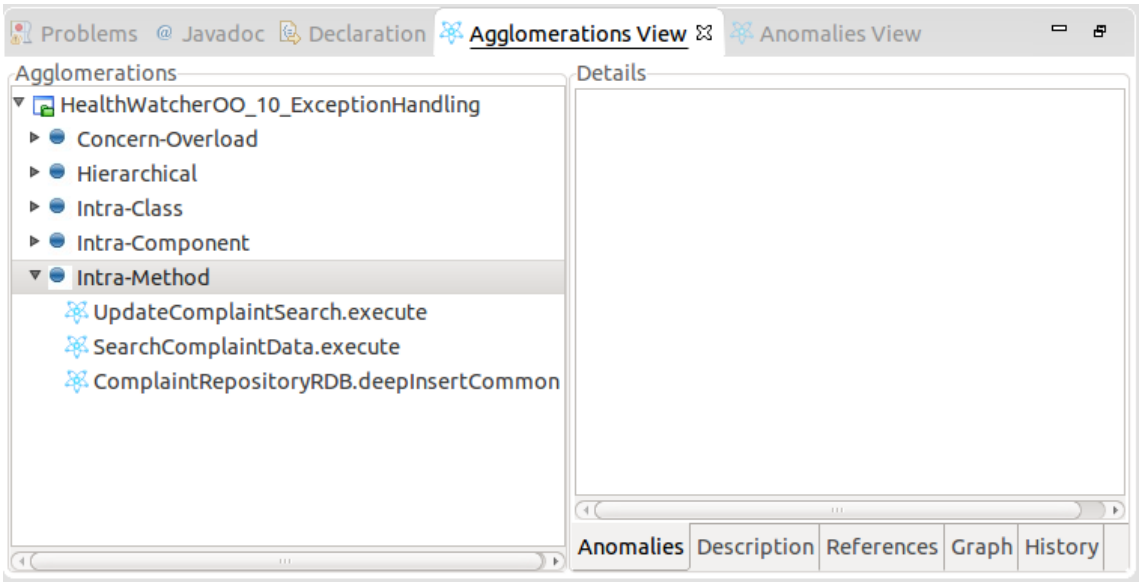

Figure 2. Intra-method agglomerations

One of the key characteristics of the synthesis technique is the provision ofinformation about agglomerations. This information is shown in the Details side of the Agglomerations View. We organize this information into five tabs: Anomalies, Description, References, Graph and History. Next we explain each of these tabs.

Anomalies

When you select an agglomeration in the left side, the Anomalies tab will show all code anomalies that take part in the selected agglomeration. In the example of Figure 3 the Anomalies tab shows two code anomalies (Dispersed Coupling and Brain Method), which are members of the *UpdateComplaintSearch.execute* agglomeration. Once you double click an anomaly in this tab, the code element affected by the anomaly will be opened in the Eclipse source code editor.

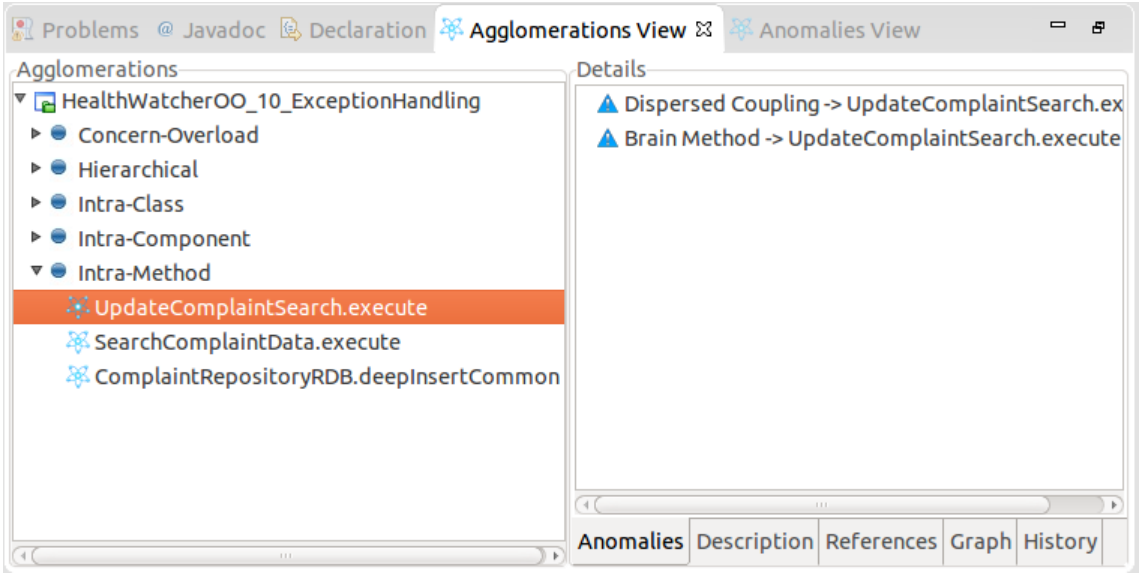

Figure 3. Anomalies Tab

Description

The Description tab shows a summarized description about the selected agglomeration. The description is composed by the following information:

Agglomeration description. A brief textual description of the agglomeration. This usually includes the type of agglomeration and the main element affected by the agglomeration. However, additional information may be provided depending on the type of agglomeration. For example, concern-overload agglomerations also provides information about their concerns.

Number of Anomalies. The number of code anomalies that are member of the selected agglomeration.

Types of Anomalies. A brief description of the types of anomalies that are member of the selected agglomeration. Note that an agglomeration may have more than one anomaly of the same type.

Figure 4 shows an example of description of a given agglomeration. The selected agglomeration is of the intra-method type and affects the *execute* method. This agglomeration contains 2 anomalies. Finally, the types of anomalies in the example are Dispersed Coupling and Brain Method.

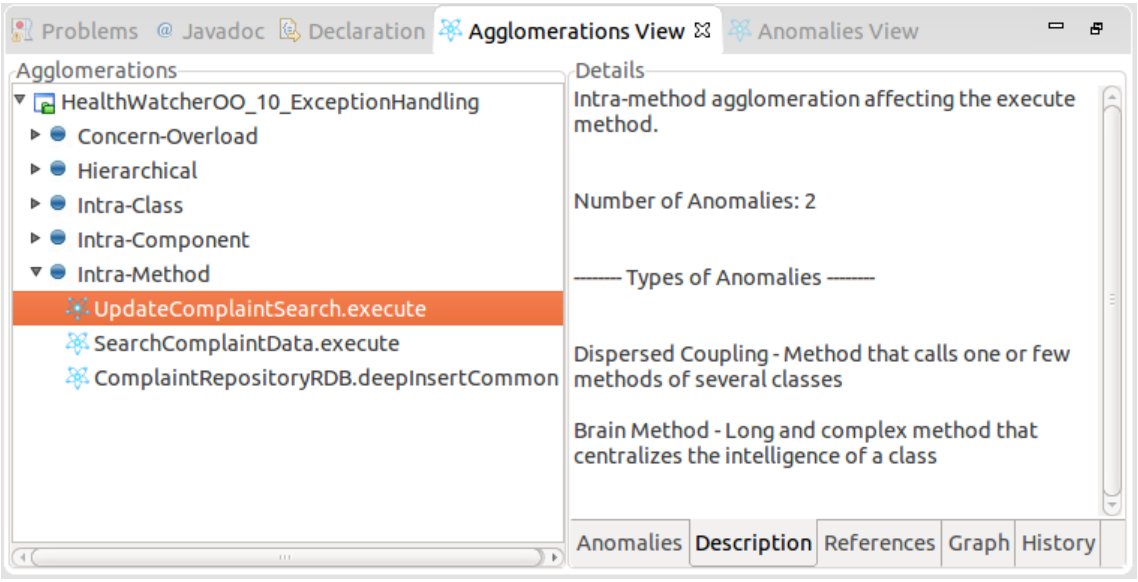

Figure 4. Description Tab

References

The References tab shows the references to each code element affected by the selected agglomeration. In the example of Figure 5 the selected agglomeration contains only one code element, which is *UpdateComplaintSearch.execute*. This code element is referenced by two code elements: *HWServlet.retry* and *HWServlet.handleRequest*. These two references are shown in the References tab. If more code elements were affected by the agglomeration, this tab would show the references to all of them.

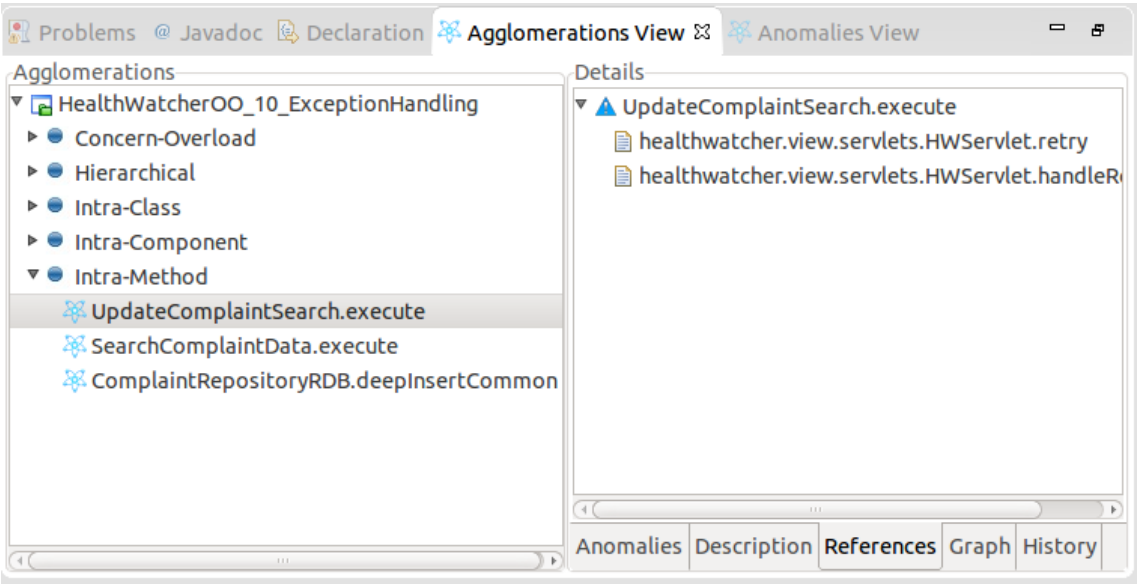

Figure 5. References Tab

Graph

The Graph tab shows a graphical representation of the selected agglomeration. This representation is **not** intended to provide a dependency graph of the agglomeration's code elements. Instead, our objective is to provide an abstract representation of the agglomeration, in order to help the developer to analyze and understand the agglomeration. Figure 6 shows an example of graph for an intra-method agglomeration. For this type of agglomeration the affected method is represented by a blue node with red borders and the code anomalies are represented by red nodes labeled with the type of each anomaly. As shown in Figure 7, additional informationwill appear when you position the mouse pointer on one of the nodes.

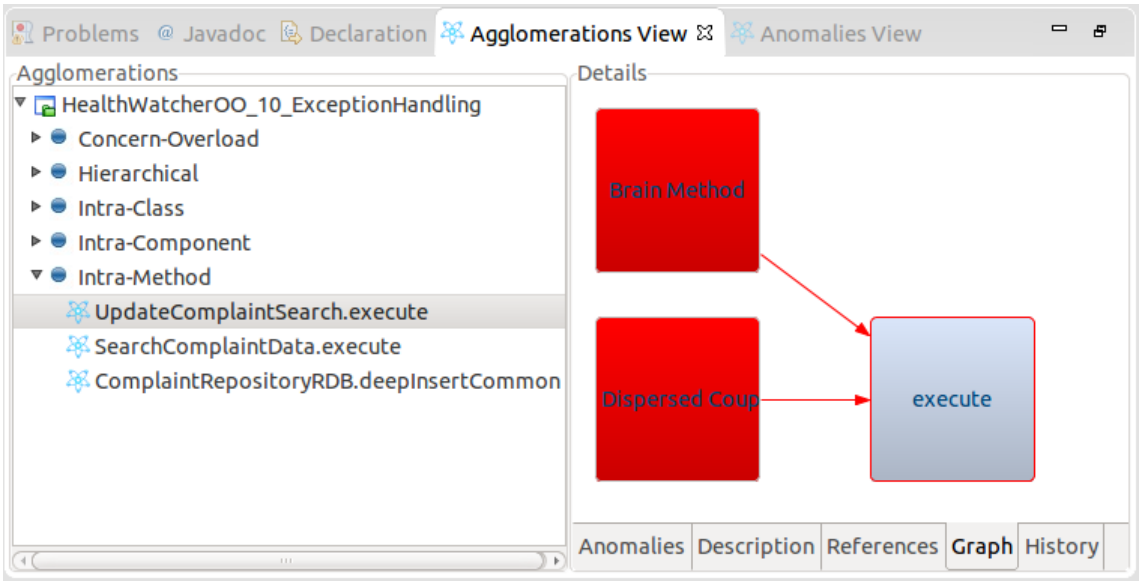

Figure 6. Graph Tab

History

The History tab shows the history of the selected agglomeration. By History we mean information about the selected agglomeration in previous versions of the target system. To be able to provide this information we firstly detect the agglomerations in previous versions of the system. After that, we store the results in files and use them to analyse the current version. The history of a given agglomeration may have 0 (zero) or many versions. Each version shows the anomalies that were member of the agglomeration. In Figure 8, for example, there are two versions: version 1 and version 5. In version 1 the agglomeration was composed by only one anomaly (Refused Parent Bequest in the *FoodComplaint* class). On the other hand, the same agglomeration was composed by three anomalies (Refused Parent Bequest in *FoodComplaint*, *AnimalComplaint* and *SpecialComplaint*) in the fifth version. Using this resource we can identify agglomerations that changed during the system's evolution.

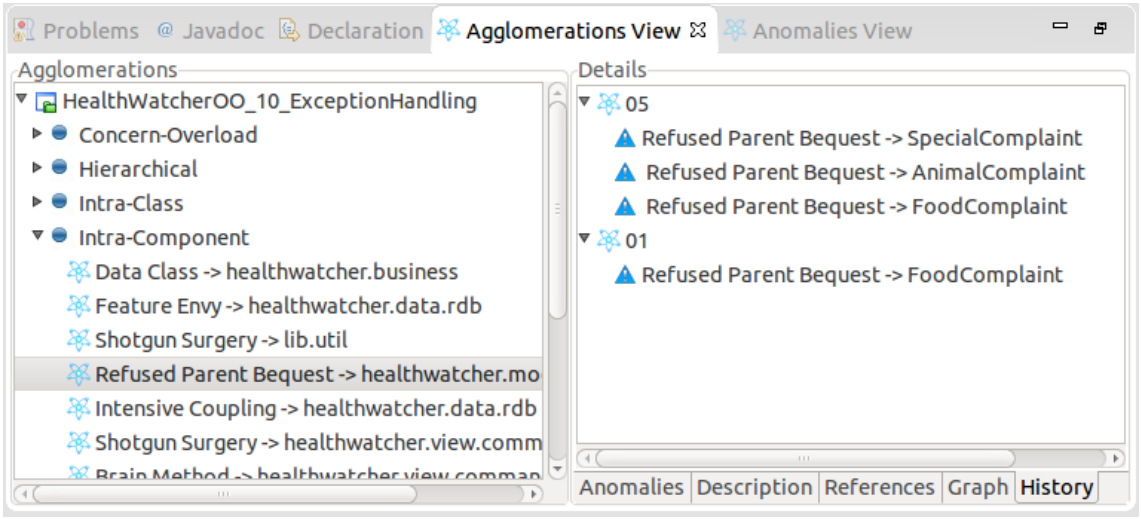

Figure 7. History Tab