Mereological Analysis of Java Programs by Using Existing Mereological Categorisations

Rezart Veliaj
National University of Maynooth, Ireland

Introduction

Mereology (from the Greek μερος, 'part') is the theory of parthood relations: of the relations of part to whole and the relations of part to part within a whole. Java is an object oriented language where every entity could be represented as an object and the presence of part to whole relations and part to part within a whole are present in different ways.

The three most common mereological relationships in Java are the binary class relationships of Association, Aggregation and Composition. These binary class relationships are very easy to represent using the modelling languages such as UML, but hard to detect in Java source code as there is a discontinuity between the source code of a program in Java and the corresponding representation in UML.

This project aims to propose an algorithm that will detect the binary class relationships in Java programs and apply the algorithm in some big open source projects. The data gathered will provide useful information on some of the best practices used to implement binary class relationships in Java. Thus the results obtained from the project will be quite useful to writing better dependable software systems.

Methods

The solution was implemented by using the ASM framework.

Figure 1 - ASM Packages (Kuleshov, 2005)

The packages are given in three layers. Layer 1 contains the core package, Layer 2 contains Tree, Commons, Util and XML packages and Layer 3 contains the Analysis package. The Core package provides the API calls to carry out core operations with the framework such as: read, write, and transform Java bytecode. Therefore the core package is sufficient and good enough to generate bytecode and make the majority of the bytecode transformations.

Procedures

The algorithm was divided in two parts: the part that does the static analysis and the part that does the dynamic analysis.

In order to proceed with the static analysis a tool was built based on the ASM framework. The tool produces csv files with the information regarding multiplicity and invocation site. Information about all the generalizations available is shown, in order to narrow down more the possible cases of bi-directional relationships.

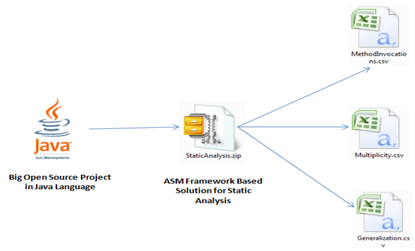


Fig 2 - Static Analysis of Java Programs

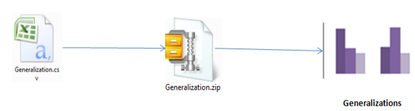


Fig 3 - Detection of Generalizations

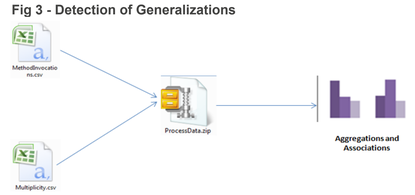


Fig 4 - Detection of Aggregations and Associations

For the dynamic analysis the exclusivity and lifetime properties needed to be detected if they hold true or false for a

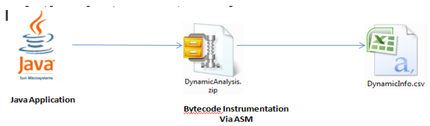


Fig 5 - Dynamic Analysis of the Java Programs

Results

5 big open source projects were taken under investigation in order to gather important data on the possible relationships between the classes in the project. The Java projects that were chosen, were the following:

- Apache Maven 3.1.1
- Junit 4.11
- Apache Ant 1.9.1
- JHotDraw 6 beta version
- Apachelvy 2.3.0

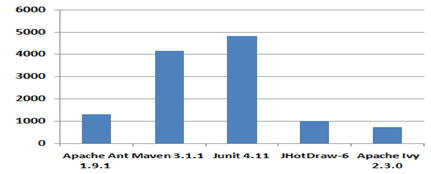


Fig 6 – Number of Classes

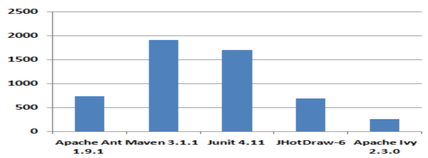


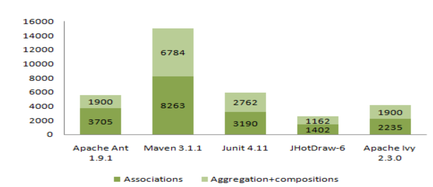
Fig – 7 Generalizations

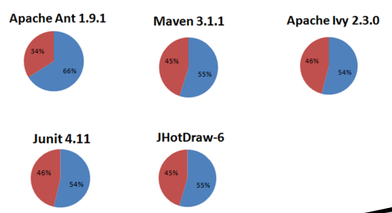
Table 1 – Association Bi-directional Relationships

Project	Number of Association Bi-directional Relationships
Apache Ant 1.9.1	49
Maven 3.1.1	50
Junit 4.11	11
JHotDraw-6	1
Apache Ivy 2.3.0	17

Table 2 - Number of association+aggregation+composition

Project	Number of Associations + Aggregations + Compositions
Apache Ant 1.9.1	5605
Maven 3.1.1	15047
Junit 4.11	5952
JHotDraw-6	2564
Apache Ivy 2.3.0	4135





Conclusion

- It seems that there is a connection between the number of classes and the number of generalizations. This means that proportionally every bigger project in terms of classes, contains in itself greater number of generalizations as well.

- In terms of association bi-directional relationships, there is a less visible connection between the number of bi-directional relationships in terms of program size, which in this case we refer to the program size, as the number of classes each program contains.

- In terms of one-directional relationships, it can be seen that there is a higher percentage of associations in relation to aggregation and compositions.

- The study was proven to be right in the case of a small example, since the relationships were detected accurately by combing both the static and dynamic analysis. represented in each project.

Acknowledgements

My special thanks to my supervisor Dr. James Power for all his assistance and guidance during the project and Dr. Rosemary Monahan for all her support and guidance during the entire Academic year at the National University of Ireland, Maynooth.

References

1. Kollmann, R., & Gogolla, M. (2001). Application of UML Associations and Their Adornments in Design Recovery. 8thWorking Conference on Reverse Engineering . Los Alamitos: IEEE.

1. Keet, C. M. (2006). Part-Whole relations in object-role models. OTM'06 Proceedings of the 2006 international conference on On the Move to Meaningful Internet Systems: AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET - Volume Part II (pp. 1118-1127). Heidelberg: Springer-Verlag

2. Kuleshov, E. (2005, 08 17). Introduction to the ASM 2.0 Bytecode Framework. Retrieved 05 31, 2013, from OW2 Consortium : http://asm.ow2.org/doc/tutorial-asm-2.0.html