Composite Annotations and Aspect-oriented Programming

author: Matej Palfi

supervised by: doc. Ing. Jaroslav Porubän, PhD.

consulted with: Ing. Milan Nosáľ, PhD.

1. Problem

```
[ @DatabaseObject
  @DatabaseType(Type="nvarchar(max)") ]
public String Name;

[ @DatabaseObject
  @DatabaseType(Type="nvarchar(max)") ]
public String Surname;

[ @DatabaseObject
  @DatabaseType(Type="nvarchar(max)") ]
public String Address;
```

- Cloned code in the form of annotations (copy->paste) - **annotations usage pattern**
- Can be dealt with so called „**composite annotations**“ replacing an annotation usage pattern with a single composite annotation

2. Cloned annotations in practice

We hypotesize that in practice there are cloned annotation usage patterns. By proving this we establish a foudation for composite annotations.

Method: Automated code analysis for detecting annotation usage patterns.

Results and contributions: We detected 199 groups of annotations in 10 projects (that makes from 19 to 20 annotation groups on a project in average). This shows that there is a significant amount of cloned annotations, thus we proved our hypothesis. This is our foundation for further research in the field of annotation usage patterns.

3. Aspect-oriented Composite Annotations

We analyse Aspect-oriented Programming as an alternative to existing composite annotations' supporting tools. We assume AOP and its inter-type declarations can be used as a standard composite annotations' definition tool.

Method: Analysis in a form of experiments of various annotation compositions in the AspectJ language.

Results and contributions: We provide a list of compositions of annotation types supported by ITD in AOP. We also discuss shortcomings of current ITD implementation with the respect to composite annotations. These results could be used by AspectJ authors to enhance ITD mechanism for better composite annotations support.