CloneDetector

Clone detector tool for Java projects

Adrian-Vlad LEP, Coordinator: conf.dr.ing. Ioana SORA

“Politehnica” University of Timisoara

Automation and Computer Engineering Faculty

Computers and Information Technology

June 2011

There are various platforms that detect copied literary or software projects.  CloneDetector, the described tool in this paper, statically examines Java source code and detects copied projects based on a set of Prolog rules. CloneDetector uses an original approach, combining the detection methods based on metrics and those based on call graphs created from the abstract syntax trees (AST).This instrument can detect type 3 clones, which are copied projects that are renamed and have additional changes on syntactic level, have added or deleted portions of code.   
 The project was developed to detect copied Java school projects. The architecture of the system is described in Figure 1. To compare the two projects we will generate from the Java code with JTransformer, an Eclipse plugin, the abstract syntax tree (AST) of each project. AST's will be formed as Prolog clauses and stored in qlf files (Quick Load File) compiled Prolog files. 1. Arhitectura sistemului
  
The analysis of two projects start at class level. We try to find for each class of the first project an equivalent class in the second one. Two classes will be considered identical if they have similarities at the class level, if their methods are similar and if the called methods and classes from the methods are similar.   
From tests we noticed that the call graph analysis (the third condition) brings a major improvement, increasing numbers of correct matches at the method level, and therefor at the class and project level.   
To meet the needs of teachers better and ease the users work, various facilities have been developed without affecting the scalability of the algorithm. CloneDetector allow the exclusion of classes, packages or folders from the analysis. This proves very useful if all projects were given pieces of code from the beginning.   
In order to analyze the projects differently depending on the given assignment, we created two profiles one loose profile and a second tighter, more exact one. These profiles can be selected at runtime and new profiles can be easily defined in the profiles.pl file.   
Also, because the method of implementation, it is relatively easy to add or delete new filters if new requirements appear or new cases want to be tested.   
 In conclusion, CloneDetector wants to be an user friendly and effective tool in detecting copied software projects. Based on the tests carried out on specific projects, we are confident in the usefulness of this platform. We noticed that programs that use strings matching algorithms, although very fast, can only detect clones of type 1, 2 and 3 if complete renames do not occur. Instruments that rely on call graphs can report all types of clones, but due to the complexity of algorithms have a higher processing time. CloneDetector can detect all clones of types 1 and 2 regardless of the number of renaming done, signaling also type 3 clones with their likelihood of being copied.