CloneAnalyzer

Forschungszentrum Informatik, Karlsruhe
Matthias Biehl, Michael Winter, Christoph Andriessens, Olaf Seng

May 13, 2005

This document is intended to support the users of the CloneAnalyzer Eclipse Plugin to install, configure and use the tool. We familiarize you with the user interface and all steps necessary to analyze your first project.

This document is not primarily a reference for developers of the CloneAnalyzer Eclipse Plugin. Developers might be interested in chapter 6 and the javadoc provided with the installation in the `doc/javadoc` directory.

Contents

Legal information

1 Introduction

1.1 What is CloneAnalyzer

CloneAnalyzer is a tool for software quality analysis. It allows you to find, display and inspect clones, which are fragments of duplicated source code resulting from lack of proper reuse. Usually the fragments are created by copy-and-paste operations. Thus information about clones might give hints to detect design flaws in software projects. For a more detailed analysis of the Cut-And-Paste AntiPattern see [1] and for a suggested refactoring see [2]. For more information about software quality analysis see `http://www.qbench.de`.

1.2 Terminology

To specify precisely what we talk about, here are some definitions for terms used in the following paragraphs.

CodeFragment consists of succeeding lines of code in a particular file. Each line is associated with a hash-value.

CloneInstance is a CodeFragment that is also an element of exactly one CloneSet

CloneSet is an equivalence class consisting of CloneInstances. All members of this CloneSet match the same ClonePattern.

ClonePattern describes a CloneSet by a list of hash-values that are associated with each line.

1.3 How you can learn about the quality of your software

CloneAnalyzer can support you in getting a quick overview of problem-prone classes containing a high amount of code duplication by providing a sortable and navigable list. By grouping CloneInstances to CloneSets and considering only the largest containing CloneInstance, this information is very compact. It can be sorted and used for navigation in the code. Within the source code visual annotations show duplicated code. The provided export function to csv-format allows further analysis on the detected clones by spreadsheet programs.

CloneAnalyzer can detect clones in projects written in any programming language since it uses a simple string comparison.

CloneAnalyzer can be used as a plugin for the Eclipse 3.0 IDE as well as on the command line. A version for Eclipse 2.1 is provided but not continued. CloneAnalyzer is written in Java and thus platform independent. So far, it has been tested on Eclipse 3.0 on Windows and Eclipse 2.1 on Windows and Linux.

2 Setup

2.1 Requirements

Currently we support a plugin for Eclipse 3.0 only. Due to incompatible changes to the Eclipse Framework it will not run under any versions prior to 3.0. Still, there is an unsupported version for Eclipse 2.1 available which has been tested under Windows and Linux.

1. Download
   - Get the file `CloneAnalyzerPluginInstall_DATE.zip`
   - The latest version of this file can be found at: `http://cloneanalyzer.sourceforge.net`
2. Copy Files
   - Close all running Eclipse instances.
   - Extract all the files in the archive to your Eclipse installation directory. There should be a new folder
     `<your installation path of eclipse>/eclipse/plugins/CloneAnalyzer`
3. *For Eclipse 2.1 only:* Finish Installation in Eclipse
   - Choose `Window` → `Customize Perspective`
   - Pick `Window` > `ShowView`, Select `CloneTreeViewer`
   - Pick `Other`, Select `CloneAnalyzerMenu` → `OK`
   - Choose `Window` → `Show Views` → `Other`
   - Pick `CloneAnalyzer` → `CloneTreeViewer` → `OK`

3 Configuration

In the file `comments.conf` you can specify comments. This file consists of a number of pairs of regular expressions. Each pair specifies the starting and ending of a comment. The regular expressions use the standard java regexp syntax. The specified comments are used for all files considered by the file filter. (see `http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html`)

Even the Comment Description File has a comment syntax: Comments for this file start with # at the beginning of the line and end at the end of the line. Empty lines are ignored.

It is located in `<your workspace>/.metadata/.plugins/CloneAnalyzer/comments.conf`. If it is not there, start Eclipse and the CloneAnalyzer Plugin, it will be copied from the default version of this file in `<your installation path of eclipse>/eclipse/plugins/CloneAnalyzer/comments.conf` Don't edit the default file, since it won't have any effects. There is a default comment description to be used as template for Java/C/C++ projects (`comments-java-c-cpp.conf`) and one for Pascal/Delphi (`comments-delphi.conf`).

4 Eclipse Plugin

4.1 CloneTreeView

After running the Build or Rebuild, you will see some entries in the CloneTreeView. On the first level there is the list of CloneSets, on the second level there is the list of CloneInstances of the corresponding CloneSet. You can browse this tree-structure and call actions on the CloneInstances from a context menu by right-clicking. Double-clicking takes you directly to the position of this CloneInstance and opens an editor. You can easily find the selected CloneInstance since its annotation is red. For each CloneSet you will be shown the length, the number of instances, and a number. The number is not guaranteed to be the same on each run, since its not to identify a CloneSet but rather to distinguish different CloneSets.

4.1.1 CloneTreeView Toolbar

You can sort the CloneSet Entries in the Tree by the magnitude of the set (number of CloneInstances) or by the length of the ClonePattern. Within the CloneSet, CloneInstances are sorted lexicographically.

Sometimes it is interesting to know the relation of the length of the ClonePattern to the total length of the containing source files; we call this the *coverage* of a CloneInstance. Since this value is specific for one CloneInstance, sorting CloneSets for coverage can be done either by the maximum or by the average coverage in the CloneSet.

If the source code is changed and saved, CloneAnalyzer will notice and the warning button will be activated. The warning button indicates, that the CloneInformation is outdated. If you click on the button, the computation is triggered and your files are analyzed using the same configuration as before.

4.2 CloneEditor

In the editor the CodeFragments of the CloneInstances are marked by a colored annotation on the left bar. CloneInstances are usually yellow, selected ones are red. You can select a CloneInstance by double-clicking on its entry in the CloneTreeView or the context menu of the editor. Within the CloneEditor do a right-click on the rulerbar or on the editor and you will get a list of all CloneInstances that occur in this line, the selected CloneInstance is marked by $\gg$. You can select one of the CloneInstances and perform an action. For example, you can locate the CloneInstance in the CloneTreeView to show all the CloneInstances belonging to the same CloneSet.

5 Console Application

There is a special version of CloneAnalyzer to use on the command line, without having to use Eclipse or its libraries.
usage: `java -cp CloneAnalyzer.jar CloneAnalyzerConsole` *options*

or: `java -jar CloneAnalyzer.jar` *options*
Where *options* stands for:

- `-m` MinCloneLength (optional Parameter): Only show CloneInstances that consist of *MinCloneLength* lines
- `-d` InputDirectory (optional Parameter): Read the files from *InputDirectory*
- `-f` FileFilter (optional Parameter): Regular expression on the absolute path describing the files to read
- `-c` CommentsFileName (optional Parameter): Path and filename of the `comments.conf`
- `-iw` IgnoreWhitespace (optional Parameter): If true, whitespace will be ignored, otherwise exact matches only
- `-ic` IgnoreComments (optional Parameter): If true, comments will be ignored, otherwise comments will be considered for clone-detection

If you don't specify some or all of the parameters, the following default configuration will be used for omitted values: `-m 15 -ic true -iw true -d . -f .*\.(java|c|cpp|pas) -c comments.conf`

6 Development

6.1 Installing the source code

First, you need to extract the source-package included in the file `sources.zip` in the installation package `CloneAnalyzerPluginInstall_DATE.zip`. Then, create a new project in Eclipse, based on the extracted files.

6.2 Developer documentation

The documentation for developers is written in javadoc. You can find it in the `doc/javadoc` directory of the source-package.

6.3 Building a distribution

If you want to build a distribution, you can use the `build.xml` ant script in the root directory of the project.

Target `backup` zips down a backup copy to the `bak` directory.

Target `dist` builds a complete distribution containing all the config files. It is written to the `release` directory.

7 FAQ

- Q: When I analyze large projects I get an OutOfMemoryError.
  A: You need to tell the Java VM to provide a larger heap.
  - On the command line replace `java` with `java -Xss8m -Xmx512m`. You might want to change the maximum heap size (`-Xmx`) according to your project and machine (see `http://java.sun.com/docs/hotspot/gc1.4.2/index.html` and `http://java.sun.com/j2se/1.4.2/docs/tooldocs/windows/java.html`)

  - For Eclipse, change the start parameters to `eclipse.exe [eclipse arguments] -vmargs -Xss8m -Xmx512m`
- Q: CloneSet numbers change
  A: The numbers of the CloneSets do not have any meaning. Their only purpose is to provide a way to distinguish different clone sets. When you run the analysis again, the same CloneSet will get a different number.
- Q: No output for a long time, excessive memory use.
  A: Your Project might be too large and contain too many clones to handle efficiently at this time. Sometimes generated code fragments are quite large. For example wrapper for COM objects might be huge (up to several MLOC). Since it is not interesting to find duplicates in generated source code, we recommend to exclude generated code from the analysis.
- Q: Comment ignorance does not work, though it is turned on (`ic true`). Why?
  A: Our pattern detection is based on regular expressions and does not consider context information. It might be tricked using escaped characters (e.g. `System.out.println(''/*'');`. We will solve this problem in the next major release. For the moment, turn the comment detection off (`ic false`).
- Q: I get strange results using the combination of `ic true` and `iw true`. What is wrong?
  A: The combination of parameters does not make sense, because the whitespace between comments is considered for clone detection. Strange behaviour may be observed, due to "invsible" whitespace patterns of tabs and blanks.

8 Problems

Please send bugs and wishlists to Matthias Biehl on `http://cloneanalyzer.sourceforge.net/contact.html`

References

[1] William J. Brown, Raphael C. Malveau, McCormick McCormick, and Thomas J. Mowbray. *Anti Patterns: Refactoring Software, Architectures and Projects in Crisis*. John Wiley & Sons, 1998. ISBN: 0–471–19713–0.

[2] Martin Fowler, Kent Beck, John Brant, Opdyke Opdyke, and Don Roberts. *Refactoring: improving the design of existing code*. Object Technology Series. Addison-Wesley, 1999.