Practical Lab Series 2 – Clone Detection

Code cloning is a phenomenon that is of both scientific and practical interest. In the lecture and the related papers, clone detection and management techniques were discussed, as well as the various arguments surrounding the problems that code cloning causes.

In this lab we will build our own clone detection and management tools. Such tools should be of help to software engineers like yourselves, so be sure that your solution will at least satisfy your own needs! Compared to Lab Series 1, this assignment will be more open. Your solution will be graded using more generic criteria, with a stronger emphasis on motivation, argumentation, evaluation and reflection. You will need to use literature discussed and referenced in the lectures to find and motivate good solutions.

Collaboration

Complete the assignment in the same group as for Series 1. You can brainstorm, but for this assignment you are not allowed to look at code from other groups or exchange solutions in detail with other groups. The Golden Rule still applies.

Assignment

In this assignment you will implement AST-based clone detection and use it to produce clones and statistics about clones for a given Java project (we use smallsql and hsqldb again). After this first step you will either:

- Implement and compare clone detection algorithms, including clones of Type II and Type III **(back-end route)**
- Implement and evaluate visualizations of clones (classes) that aid software maintenance **(front-end route)**

In both cases you will have to present a design of your solution (algorithms or visualisations), draw from existing literature for your design, describe the implementation of your solution, and provide a thorough evaluation and reflection on your design. These aspects are to be covered in a written report and in a presentation in the last week of the course.

The assignment consists of two main deliverables. Some parts are only required for the chosen route (where indicated):

1. Working prototype implementation of a clone management tool, consisting of the following elements:
   (a) An AST-based clone detector whose back-end is written in Rascal that detects at least Type I clones in a Java project:
      - Detected clone classes are written to a file in a textual representation.
      - Clone classes that are strictly included in others are dropped from the results (subsumption).

    - The detector is scalable and works on bigger projects such as hsqldb.
  - (b) A report of cloning statistics showing at least the % of duplicated lines, number of clones, number of clone classes, biggest clone (in lines), biggest clone class (in members), and example clones.
  - (c) (**front-end route**) Insightful visualizations of cloning in a project that help with software maintenance. The lecture discusses several example visualizations you could use.
  - (d) (**back-end route**) A benchmark Java project that serves to demonstrate the correctness of your clone detection algorithms with respect to the types of clones they are meant to detect.
2. A written report that (1) describes, (2) motivates, and (3) reflects on the following elements (not in order):
  - (a) An explanation of the implementation of your clone detection algorithm(s).
  - (b) (**front-end route**) The requirements your tool satisfies from the perspective of a maintainer (see for instance [33]), and the related implementation choices.
  - (c) (**front-end route**) The implementation of your visualization(s) and a reflection that establishes to which extent the requirements have been satisfied by your visualizations
  - (d) (**back-end route**) The exact type of clones your tool detects, giving a sufficiently detailed definition that enables critical assessment of your benchmark.
  - (e) (**back-end route**) A detailed discussion of the (differences in) cloning statistics produced by your different algorithms.

To score higher grade than the base grade (7) additional work needs to be done. For example, implementing many algorithms or visualisations of different kinds or doing work from the other route. For details, see Table 8.

Implementation

The clone detection algorithms must be in implemented in pure Rascal. The visualisations can be written in other languages.

Related Work on Software Clones

The following resources can help you get acquainted with clone management:

- I. Baxter et al. “Clone detection using abstract syntax trees”. In: *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272).* 1998, pp. 368–377. DOI: `10.1109/ICSM.1998.738528`
- C. K. Roy, J. R. Cordy, and R. Koschke. “Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach”. In: *Sci. Comput. Program.* 74.7 (May 2009), pp. 470–495. ISSN: 0167-6423. DOI: `10.1016/j.scico.2009.02.007`

– D. Rattan, R. Bhatia, and M. Singh. "Software Clone Detection: A Systematic Review". In: *Information and Software Technology* 55.7 (July 2013), pp. 1165–1199. ISSN: 0950-5849. DOI: `10.1016/j.infsof.2013.01.008`.
– C. K. Roy, M. F. Zibran, and R. Koschke. "The Vision of Software Clone Management: Past, Present, and Future (Keynote paper)". In: *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on.* Feb. 2014, pp. 18–33. DOI: `10.1109/CSMR-WCRE.2014.6747168`
– C. Kapser and M. W. Godfrey. "'Cloning Considered Harmful' Considered Harmful". In: *2006 13th Working Conference on Reverse Engineering.* Oct. 2006, pp. 19–28. DOI: `10.1109/WCRE.2006.1`
– A. Hamid and V. Zaytsev. "Detecting Refactorable Clones by Slicing Program Dependence Graphs". In: *Post-proceedings of the Seventh Seminar on Advanced Techniques and Tools for Software Evolution, SATToSE 2014, L'Aquila, Italy, July 9–11, 2014.* Ed. by D. di Ruscio and V. Zaytsev. Vol. 1354. CEUR Workshop Proceedings. CEUR-WS.org, 2014, pp. 37–48. URL: `https://dare.uva.nl/search?identifier=d0ad3c4a-5d65-44d7-bbe9-2c062598c64b`

The first three are overviews [30, 28, 29], the next one is a highly cited controversial piece [17], the last one is an example paper that can result from a Master's thesis – it is easy to read and contains a simplified brief overview of the field [11].

Related Work on Visualization

Papers:

– H. Murakami, Y. Higo, and S. Kusumoto. "ClonePacker: A Tool for Clone Set Visualization". In: *Proceedings of the 22nd International Conference on Software Analysis, Evolution and Reengineering.* Ed. by Y.-G. Gueheneuc, B. Adams, and A. Serebrenik. IEEE, 2015, pp. 474–478. ISBN: 978-1-4799-8469-5. DOI: `10.1109/SANER.2015.7081859`
– L. Voinea and A. C. Telea. "Visual Clone Analysis with SolidSDD". in: *Proceedings of the Second IEEE Working Conference on Software Visualization.* IEEE, 2014, pp. 79–82. DOI: `10.1109/VISSOFT.2014.22`
– A. Hanjalic. "ClonEvol: Visualizing Software Evolution with Code Clones". In: *Proceedings of the First IEEE Working Conference on Software Visualization.* IEEE, 2013, pp. 1–4. DOI: `10.1109/VISSOFT.2013.6650525`

Visualization Libraries:

– Salix is a library for interactive tools and visualizations in RASCAL using a browser[12]. Several demos are available, including live programming of state machines, similar to the running example of [32]. – powerful yet experimental

[12]`https://github.com/cwi-swat/salix`

– Examples of external visualisation libraries are D3[13], vis[14], Vega[15] and Gephi[16].

2.1 Related Work on Benchmarks

– K. Jezek and J. Dietrich. “API Evolution and Compatibility: A Data Corpus and Tool Evaluation”. In: *Journal of Object Technology* 16.4 (Aug. 2017), 2:1–23. ISSN: 1660-1769. DOI: `10.5381/jot.2017.16.4.a2`

Grading

Series 2 is primarily assessed on the report you submit. You will also give a mandatory presentation that complements your report and can influence your grade. Your submission includes a PDF file of your report, the source-code of your implementation and the textual output reports produced for smallsql and hsqldb. The files are checked for plagiarism automatically. You will not receive a grade if you do not support a report or the source code, or if you do not give a presentation. The presentation should describe the design of your solution(s) and details the process and results of your evaluation. The presentation is a joint presentation between both members of the group.

To qualify for grading you first need a solution that complies to the assignment as described. The base grade is 7 and the conditions laid out in Table 8 modify the grade.

Deadline

The deadline for handing in your submission is given on canvas. Shortly after the deadline, the presentation sessions will be held according to a schedule announced in due course.

[13] `https://d3js.org`
[14] `http://visjs.org`
[15] `https://vega.github.io/vega/`
[16] `https://gephi.org`

Condition	**Max grade modifier**
(**implementation**) Type I clone classes are incorrectly detected.	-1.0
(**implementation**) Missing textual output reports (including statistics) for smallsql or hsqldb	-0.5
(**report**) Type I algorithm is described incompletely or incomprehensibly.	-1.0
(**report**) Unfounded, unsupported, or illogical motivations for the design	-1.0
(**back-end route**) Benchmark is limited in scope or not thorough.	-1.0
(**back-end route**) Comparisons between implemented algorithms is lacking or superficial.	-1.0
(**front-end route**) Cloning visualizations do not give insight or do not work properly.	-1.0
(**front-end route**) Requirements are not clearly defined or not sufficiently reflected upon.	-1.0
(**front-end route**) Correct detection of Type II, III, or IV clone classes	+1.0
(**back-end route**) Correct detection of Type IV clone classes	+1.0
(**presentation**) Well-motivated design of solution(s) presented clearly	-0.5 to +0.5
(**presentation**) Sound evaluation of solution(s) presented clearly	-0.5 to +0.5
(**presentation**) Insightful discussion in response to questions	-0.5 to +0.5
You have designed, implemented, and evaluated additional solutions	+2.0

Table 8: Grading Conditions and Scoring for Series 2