Code clones thesis proposal

14. februar 2022

Motivation

- Modern clone detection tools are as far as I've seen not very suitable for use while programming, but rather for analysis after code has been written. It seems likely that managing these clones when analyzed later is harder than managing code clones detected in real-time.
- One plugin for Eclipse exists already, but the plugin does not have functionality for merging clones.
- No tools I've seen provide "real-time" clone detection and functionality to merge them.

Proposal

I want to develop a tool which provides real-time clone detection and functionality to merge the detected clones in a modern IDE. This could be suitable to implement as an LSP server, making it an IDE agnostic tool.

Potential research questions

- How could real-time detection of code clones affect code quality compared to later analysis?
- How should real-time code clone detection be implemented when focusing on efficiency and scalability?
- How should real-time code clone merging be implemented when focusing on efficiency and scalability?
- How should code clone detection be offered to IDE users? (LSP, plugins, ...)

Things to research and challenges

- Real-time clone detection (while typing in an IDE)
- Refactoring-Oriented clones
- Code clone merging
- Incremental code clone detection?
- Efficient detection of new code clones in a changing codebase
- Providing clone detection as an LSP server or plugin

Previous work / competition

- SimEclipse: An Eclipse plugin. `https://clones.usask.ca/pubfiles/articles/UddinIntegratedCM_CSACON-2015.pdf`

Tools of relevance

- SimLib: a clone detection engine/library written in Java `https://github.com/suddin/ca.usask.cs.srlab.simLib`
- lsp4j: A Java library for creating LSP servers `https://github.com/eclipse/lsp4j`

Current title in mind

- "Towards modern support for clone management in IDE's"
- "Real-time management of code clones in an IDE environment"

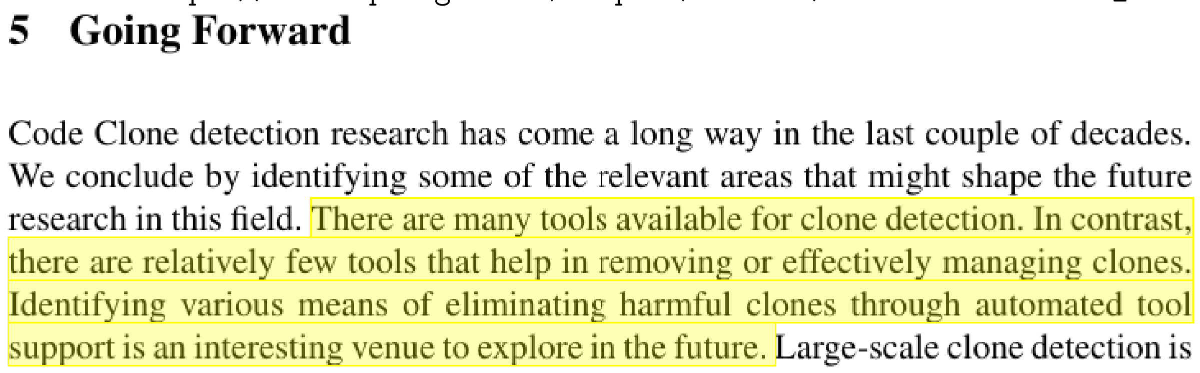
Going Forward

Going forward I will for the rest of the semester focus on:

- Getting an overview of existing techniques for both detection and merging of clones
- Determine if LSP is an appropriate protocol for implementing this tool
- Read more about general refactoring and software quality for the essay

Relevant points from articles I've read

From `https://link.springer.com/chapter/10.1007/978-981-16-1927-4_4`

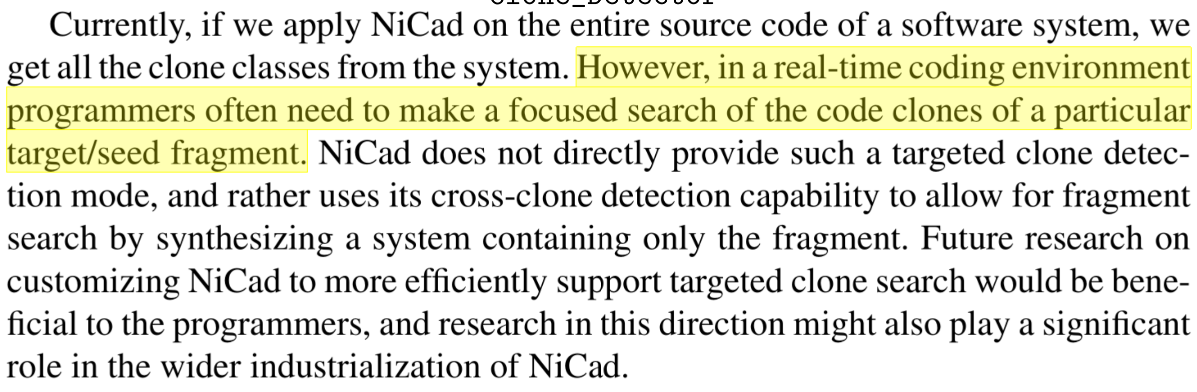


5 Going Forward

Code Clone detection research has come a long way in the last couple of decades. We conclude by identifying some of the relevant areas that might shape the future research in this field. There are many tools available for clone detection. In contrast, there are relatively few tools that help in removing or effectively managing clones. Identifying various means of eliminating harmful clones through automated tool support is an interesting venue to explore in the future. Large-scale clone detection is

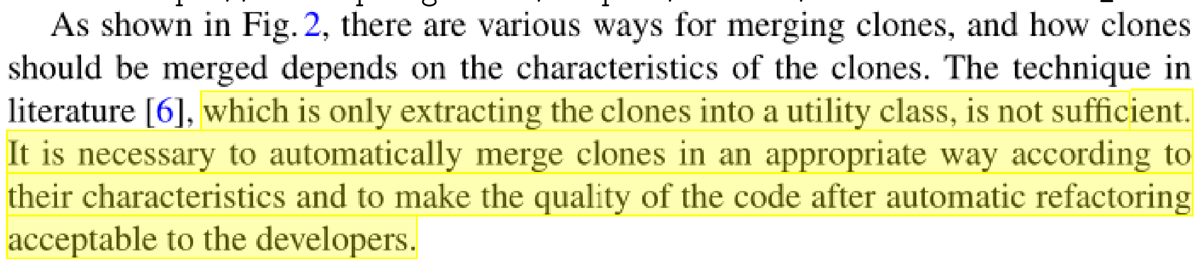
---

From `https://www.researchgate.net/publication/353661146_NiCad_A_Modern_Clone_Detector`



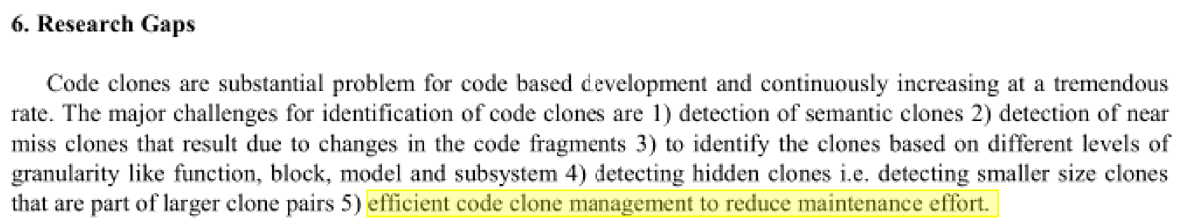
Currently, if we apply NiCad on the entire source code of a software system, we get all the clone classes from the system. However, in a real-time coding environment programmers often need to make a focused search of the code clones of a particular target/seed fragment. NiCad does not directly provide such a targeted clone detection mode, and rather uses its cross-clone detection capability to allow for fragment search by synthesizing a system containing only the fragment. Future research on customizing NiCad to more efficiently support targeted clone search would be beneficial to the programmers, and research in this direction might also play a significant role in the wider industrialization of NiCad.

From `https://link.springer.com/chapter/10.1007/978-981-16-1927-4_13`



As shown in Fig. 2, there are various ways for merging clones, and how clones should be merged depends on the characteristics of the clones. The technique in literature [6], which is only extracting the clones into a utility class, is not sufficient. It is necessary to automatically merge clones in an appropriate way according to their characteristics and to make the quality of the code after automatic refactoring acceptable to the developers.

From `https://www.sciencedirect.com/science/article/pii/S1877050918308123`



6. Research Gaps

Code clones are substantial problem for code based development and continuously increasing at a tremendous rate. The major challenges for identification of code clones are 1) detection of semantic clones 2) detection of near miss clones that result due to changes in the code fragments 3) to identify the clones based on different levels of granularity like function, block, model and subsystem 4) detecting hidden clones i.e. detecting smaller size clones that are part of larger clone pairs 5) efficient code clone management to reduce maintenance effort.