# Code Clone Detection Experience at Microsoft

Yingnong Dang, Song Ge, Ray Huang and Dongmei Zhang
Microsoft Research Asia
yidang;songge;rayhuang;dongmeiz@microsoft.com

## ABSTRACT

Cloning source code is a common practice in the software development process. In general, the number of code clones increases in proportion to the growth of the code base. It is challenging to proactively keep clones consistent and remove unnecessary clones during the entire software development process of large-scale commercial software. In this position paper, we briefly share some typical usage scenarios of code clone detection that we collected from Microsoft engineers. We also discuss our experience on building XIAO, a code clone detection tool, and the feedback we have received from Microsoft engineers on using XIAO in real development settings.

## Categories and Subject Descriptors

D.2.7 [Software Engineering]: Distribution, Maintenance, and Enhancement – *Restructuring, reverse engineering, and reengineering*

## General Terms

Management

## Keywords

Clone detection, Experience

## 1. INTRODUCTION

Reusing code fragments via copy-and-paste, with or without modifications or adaptations, is called code cloning, which is a common behavior of software engineers for quick code reuse [4]. In general, the number of code clones is likely to increase as the scale of code bases increases [2]. In this paper, we briefly share our experience on the design and development of XIAO, a code clone detection tool serving the needs of Microsoft engineers, as well as some initial feedback we obtained from Microsoft engineers after they used XIAO. We believe that our experience may help the community better understand the practical usage of code clone detection tools in the development process of large-scale commercial software.

## 2. CODE CLONE DETECTION IN THE SOFTWARE DEVELOPMENT PROCESS

We summarize as follows the three typical usage scenarios of clone detection in the software development process that we have collected at Microsoft.

**Fix Bugs Once** If a bug is identified in a piece of code with duplicated copies, it is desirable to have the ability to fix all of them at once. This scenario is beneficial to multiple stages of the development process as long as there are bug fixing tasks; for example, during the feature implementation stage, stabilization stage and post-release maintenance stage.

**Footprint Reduction** Code clones can be found at various degrees for different product teams we have worked with in Microsoft. Some teams are keen on reducing the memory footprint of their components; they look for every possible opportunity to achieve this goal. Removing code clones is one of the important actions they want to take.

**Clone Quantity Monitoring** The quantity of code clones could be one of the metrics that measure the quality of a code base to prevent unnecessary code clone creation. Running code clone detection tools periodically and monitoring the overall clone quantity may serve this purpose. One possible metric is the percentage of cloned LOC (lines of code) compared with the total LOC of a code base.

## 3. XIAO

Based on the usage scenarios discussed in Section 2, we identified the following key requirements for XIAO:

**Near-miss Code Clone Detection** The effectiveness of detecting near-miss code clones [3] is one of the key challenges to identify inconsistencies between cloned copies and enable the fix-bugs-once scenario.

**Scalability** Scalability is a basic requirement for a practical code clone detection tool, since the number of lines of code for commercial software is ever increasing [4]. Not surprisingly, the sizes of code bases for many Microsoft products are very large scale.

**Usability** Development teams have limited time and resources. It is difficult for them to adopt a tool that is not flexible and has a steep learning curve. Therefore, it is important for the clone detection tool to be easily customizable and intuitive to use.

**Easy Deployment** Different product teams may have similar, but not exactly the same, development environments including build systems, source management tools and bug management tools. It should have a zero or low cost for engineers to deploy a code clone detection tool in their work environments, with the tool working seamlessly with existing tools.

Based on these requirements, we developed XIAO, a code clone detection tool. XIAO has its own light-weight parser, so it does not depend on any compilers. This allows XIAO to be easily deployed in various build environments. Currently, the parser supports C/C++ and C# languages, which serves the needs of a good number of teams in Microsoft. In addition, Xiao provides an extensible framework which allows users to plug in their own parsers into the system to support other languages. The core clone detection algorithm of XIAO is able to effectively detect near-miss code clones [1]. XIAO's clone detection algorithm is implemented such that it can run in both a single machine environment and Microsoft's High Performance Computing clusters **Error! Reference source not found.**. XIAO's frontend provides an intuitive way to visualize the differences between two clone copies. It also facilitates users to efficiently explore the code clone detection results.

## 4. EXPERIENCE WE LEARNED

We released the first version of XIAO to Microsoft engineers in April 2009. There were more than 750 downloads of the tool as of 12/31/2010. The size of code bases that XIAO is used to detect code clones varies from several thousand lines to over 50 million.

Engineers from different teams in Microsoft have used XIAO in their development process. For example, one developer from an application software development team used XIAO to identify potential bugs caused by inconsistent code clones and to find refactoring candidates. He reviewed 69 clone groups with 180 cloned snippets in total and found that about 10% have potential code defects caused by inconsistent code clones, and 33% of them could be refactored. Another developer used XIAO to reduce the footprint of a component with more than 200K LOC. As a result, the percentage of cloned LOC was reduced from 30% to 25%. A test team developed a plug-in parser to parse the language used in their product and used the tool to find duplicated test code and then refactor it.

Based on our extensive communications with XIAO users at Microsoft, we summarize our experience and learning as follows.

*Detecting Near-Miss Code Clones*
Code cloning may occur inside one component and even cross components. Different cloned copies may be owned by different developers. It is quite likely that when one developer updates his/her copy for bug fixes, the other owners of the cloned copies may not be aware of the changes. The consequence of such inconsistent updates can result in near-miss code clones. We have seen a great number of such scenarios when we worked with different development teams. Therefore, it is important for a code clone detection tool to reliably detect near-miss clones.

*Flexible Filtering*
In general, different teams have different requirements on which part of the code base they want to run the clone detection tool. For example, automatically generated code, prototype code, dead code and test code are not usually required for clone detection. Furthermore, some teams may require that certain functions and statements be excluded from clone detection; for example, assertions and log printing code for debugging purposes. All of these requirements help reduce the number of clones that are not of interest or are meaningful to development teams. Using string patterns, XIAO provides a flexible filtering mechanism to exclude folders, files, functions and statements that should be ignored during clone detection.

*Intuitive and Efficient Graphical User Interface (GUI)*
We received a lot of feedback on the usability of XIAO, and in particular XIAO's GUI for code clone exploration and visualization. We released several versions of the tool, mainly focusing on improving the usability based on the user feedback.

- Efficient Exploration. There can be a great number of clones found in large-scale code bases. Therefore, it is important to enable users to explore the clones easily and efficiently. Engineers have a need to sort and filter clones based on different metrics, e.g., source tree structure, size of cloned snippets, size of a clone group, size of difference between different copies, etc. The functionalities of XIAO's frontend meet these needs well.
- Intuitive Visualization of Clone Differences. The types of differences between cloned copies can be diverse; engineers want to identify the most important differences first. This will help them quickly find out whether there are any potential bugs in the cloned code or whether the cloned code needs to be refactored. XIAO classifies the clone differences into four categories and uses visualization techniques to enable users to view the near-miss code clones intuitively.
- Tagging. Not all detected clones are of interest to the engineers. For example, some clones are by-design and they will not be refactored. It is difficult to automatically filter out the by-design clones because deep domain knowledge may be required. XIAO provides a tagging functionality for users to tag code clones in three categories: immune (uninteresting), problematic inconsistencies and refactoring opportunities. The tagging results are also beneficial for further research on clone related analysis.

*Integrated Solution*
Nowadays engineers are already working with a number of different tools to complete their daily tasks. There is a cost for them to adopt any additional tools. In order to enable wide adoption, we built a rich set of features to make XIAO seamlessly work with existing development tools. For example, when engineers decide to take action on a detected clone, XIAO's GUI allows them to easily open a bug in the bug management system they use. Another example is the code clone statistic report that can be easily shared with team members to facilitate clone quantity monitoring.

## 5. SUMMARY

In this paper, we shared our experience on building XIAO, a code clone detection tool, and its usage at Microsoft.

From working with engineers in real development settings, we learned that it is important for clone detection tools to reliably detect near-miss clones and to effectively reduce the number of clones of no interest. In addition, how to present the detected clones also has impact on clone understanding and the action taken by the engineers. We hope that the sharing of our experience calls for more research on these directions.

## ACKNOWLEDGEMENT

We thank the Microsoft engineers and researchers for their usage of XIAO and their valuable feedback and discussion. We thank Prof. Sunghum Kim and Katsuro Inoue for the valuable comments.

## REFERENCE

[1] Y. Dang, S. Ge, Y. Qiu and D. Zhang: XIAO: Effective Near-Miss Code Clone Detector, *Microsoft Technical Report 2011*, to appear.

[2] M. Gabel, J. Yang, Y. Yu, M. Goldszmidt, and Z. Su. Scalable and Systematic Detection of Buggy Inconsistencies in Source Code. In *OOPSLA 2010* (SPLASH 2010).

[3] H. Kim, Y. Jung, S. Kim and K. Yi, "MeCC: Memory Comparison-based Clone Detector", In ICSE 2011.

[4] R. Koschke. Survey of research on software clones. In *Proc. Duplication, Redundancy, and Similarity in Software*, 2007.

[5] http://technet.microsoft.com/en-us/hpc/default