

Title	Detection and evolution analysis of code clones for efficient management of large-scale software systems
Author(s)	Choi, Eunjong
Citation	大阪大学, 2015, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/52021
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

Form 3

Abstract of Thesis

Name (Eunjong Choi)	
Title	Detection and evolution analysis of code clones for efficient management of large-scale software systems (大規模ソフトウェアの保守効率化のためのコードクローン検出・進化分析)

Abstract of Thesis

In recent decades, large-scale software systems have become mainstream.
Such software systems have complicated the maintenance process by increasing efforts such as inspection and understanding of the existing source code.
Therefore, to maintain these systems, a great deal of work and time are necessary.
To alleviate this problem, this research focus on a well-known factor hindering the software maintenance task, a code clone
(i.e., a code fragment that has other code fragments identical or similar to it in the source code).
It is widely believed that code clones complicate software maintenance.
For example, when changes to code clones in a clone set (i.e., a set of code clones that are identical or similar to each other) are inconsistent, the developer needs to identify inconsistently changed code clones and apply consistent changes to them.

Thus far, many tools and techniques have been proposed for supporting the detection and management of code clones.
However, most are insufficient for supporting code clone related tasks during the software maintenance process for large-scale software systems.
To resolve this problem, this study attempts to solve two important problems that code clones face.
That is, ``Which type of normalization dose make code clones to detected with high speed from large-scale software systems? '' and ``Which supports are necessary for more widely used tools that support clone refactoring? ''.

To solve the first problem, this research proposes six approaches for detecting code clones with preprocessing input source files using different degrees of normalizations (e.g., the removal of white spaces, tokenization, and the regularization of identifiers).
More precisely, each type of normalization is applied to the input source files, and equivalence class partitioning of the files is then conducted during the preprocessing.
Code clones are then detected from a set of files that are representatives of each equivalence class using a token-based code clone detection tool called CCFinder.
The proposed approaches can be categorized into two types, an approach with non-normalization and approaches with normalization.
The former type is the detection of only identical files without normalization, whereas the latter category is the detection of identical files with different degrees of normalization such as the removal of all lines containing macros in C program.
From a case study, it was observed that the detection times of the proposed approaches are at least two-times faster than an approach that uses only CCFinder.
It was also found that the approach with non-normalization is the fastest of the proposed approaches for many cases.

To resolve the second problem, this research presents an investigation of clone refactoring (i.e., merging code clones into a new method) carried out during the development of open source software systems for promoting the development of refactoring tools that can be more widely utilized.

In this investigation, it was identified that a ``Replace Method with Method Object'' is the most frequently used refactoring pattern for clone refactoring. Moreover, this research discovered that merged code clone token sequences and the differences in the token sequence lengths vary for each refactoring pattern.

様式７

論文審査の結果の要旨及び担当者

氏　名（　崔　恩瀞　）		
	（職）	氏　名
論文審査担当者	主 査　教授	井上　克郎
	副 査　教授	増澤　利光
	副 査　教授	楠本　真二

論文審査の結果の要旨

近頃，大規模のソフトウェアシステムの開発がソフトウェアの主流になっている．しかし，大規模のシステムは膨大なソースコードを含むため，それらのシステムを保守するためには多くの作業や時間が必要である．この問題を解決するために，本論文では，ソフトウェア保守における代表的な問題の１つであるコードクローン（ソースコード中に存在する同一または類似した部分を持つコード片）に着目している．

一般的に、コードクローンの存在はソフトウェアの保守を困難にするといわれている．たとえば、あるクローンセット（互いに類似したコードクローンの集合）内に一貫して変更されてないコードクローンが存在する場合，開発者はそれらのコードクローンを識別し，一貫した変更を行う必要がある．今まで，コードクローン検出や管理を支援するために，多くのツールが提案されてきた．しかし，これらのツールは大規模ソフトウェアシステムでのコードクローンの検出や管理を支援するためには不十分である．この問題を解決するために，本論文では，コードクローンが大規模システムで直面している２つの大きな問題を設定し，解いた．１番目の問題は「どのような（プログラムの要素を変換する）正規化手法を用いると大規模システムでのコードクローン検出を高速に行えるか？」であり，２目番は「（１つのクローンセット内のコードクローンを一つの単位にまとめる）クローンリファクタリングを支援するツールには，どのような機能が必要か？」である．

１番目の問題を解決するために，本論文では入力ソースコードに異なる正規化（空白の除外，識別子の正規化等）を適用し，コードクローンを検出する６種類の手法を提案・評価している．まず入力ソースコードに各正規化を適用した後に，それらのファイルを同値クラスに分ける．その後，各同値クラスから１つのファイルを選択し，選択されたファイルの集合に対してコードクローン検出ツールCCFinderを適用してコードクローンを検出する．提案手法全体として「正規化を行わない手法」と「異なる正規化を行う手法」という二つの手法に分け，前者は入力ファイルに何の正規化を適用せずに，完全一致ファイルに基づいてコードクローンを検出する．後者は，異なる正規化を入力ファイルに適用し，コードクローンを検出している．評価実験では，３つのOpen Source Software（OSS）システム対して，提案手法と，CCFinderのみを使ってコードクローンを検出する方法を適用した．その結果，CCFinderのみを使ってコードクローンを検出する方法より提案手法の方が約２倍以上早く検出できることがわかった．また，提案手法の中では，正規化を行わない手法が，より早くコードクローンを検出できることが分かった．

２番目の問題を解決するため，本論文は，OSSシステムの開発中に実施されたクローンリファクタリングを識別し，それが実施されたコードクローンの特徴を調査した．その結果，「Replace Method with Method Object」パターンが一番多く使われたことが明らかになった．また，トークン列の長さが異なるコードクローンがクローンリファクタリングの対象になることも分かった．この調査結果に基づいて，クローンリファクタリング支援ツールの提案も行っている．

これらの研究成果は，ソフトウェア工学分野において有用なものであり，新規性を持つものである．よって，博士（情報科学）の学位論文として価値のあるものと認める．