要 旨

任意のプログラム言語に自動適応する重複コード検出ツールの開発

今井 達朗

ソースコード中の複数箇所に存在する同一，もしくは類似したコードの一部を重複コードと呼ぶ．もし修正が必要なコードの一部が重複コード上に見つかった場合，その重複コードと対応するコードは同様の修正を行うべき可能性が高く，修正を検討する必要がある．

これまでに重複コードを自動的に検出するさまざまな手法が提案されている．その手法の一つにソースコード中の字句の列に基づいて重複コードを検出する手法があり，その代表的なツールとして CCFinderX がある．

しかし，これまでに提案されている重複コードを検出するツールは対象とする言語が決められており，その言語以外はツールを改造しない限り重複コードを検出できない．そこで本研究では任意のプログラム言語に自動適応する重複コード検出手法を提案する．提案手法では，前処理でソースコードをトークン列にした後，出現するトークンのうち出現数の上位いくらかを予約語，それ以外を識別子と判定する．適切な予約語数を選ぶことで，識別子は異なるが意味的に近いコード同士を重複コードとして検出できると考えられる．CCFinderX と提案手法の出力の比較を行い，適切な予約語数を調べた．また，CCFinderX で検出した重複コードはほとんど提案手法でも検出できることが目視により確認できた．一方，提案手法では検出する必要のない実用的ではない重複コードを検出することもわかった．さらに，適切な予約語数がソースコード中のトークンの種類数と比例関係にあるのではないかという仮説について検討した．その結果，実験対象としたオープンソースソフトウェアにおいては比例関係は見られなかった．

キーワード　重複コード，ソフトウェア保守，CCFinderX

Abstract

Development of a duplicate code detection tool that automatically adapts to arbitrary programming languages

Tatsuro IMAI

We call a part of the same or similar code that exists in two or more places in the source code the duplicate code. If some problems are found on the duplicate code, it is likely to make the same modifications on all of the duplicate code.

Methods for automatically detecting duplicate code have been proposed so far. As one of the methods, there is a technique to detect a duplicated code based on the sequence of tokens in a source code. CCFinderX is the typical implementation of that technique.

However, duplicate code detection tools proposed so far have a fixed input language and cannot detect duplicate code of other languages unless we do not modify the tool. Hence, in this study, we propose a duplicate code detection tool that automatically adapts to arbitrary programming languages. This tool preprocesses the sequence of tokens in a source code, and it considers the tokens with high frequency to be reserved words. It is expected that we can detect semantically similar codes as a duplicate code by choosing the appropriate number of words considered to be reserved words. We compared the output of CCFinderX and the proposed method and examined the appropriate number of reserved words. Moreover, we confirmed that the proposed method can detect most of the duplicate codes that was detected by CCFinderX. On the other hand, we found that the proposed method also detected impractical, needless

duplicate codes. Furthermore, we examined a hypothesis that an appropriate number of reserved words has a correlation to the number of the different tokens in a source code, and we could not find such correlation in the open-source softwares used in an experiment.

key words duplicate code, software maintenance, CCFinderX