コードクローン情報をソースコードに付加する保守支援ツールの提案

2020SE008 樋口匠　2020SE041 永田大和
指導教員：井上克郎

1　はじめに

保守を困難にする一つの要因として，コードクローンがある．コードクローンは，一か所に変更を加えた際に他の同形コード片にも修正が必要かどうかを検討しなければならない．しかし，膨大なソースコードの中からコードクローンを見つけ出すのは極めて困難である．この課題を解決するために多くのコードクローン検出ツールが開発されている [3].

一方，検出されたコードクローンを管理するツールも提案されているが，既存のツールには様々な課題がある．本研究では，保守作業の効率化のため, ソースコードだけで直観的にコードクローン情報を確認することができることを目指す．本研究では，コードクローンに関する情報をソースコード中にコードクローン情報を付加するツールと，VScode の拡張機能として，コードクローン情報を隠す機能，コードクローン部分にハイライトをする機能，クローン ID を利用しパス情報まで遷移する機能を開発する．

2　研究の背景

2.1　コードクローン

コードクローンとは, 同一または類似したソースコードの集合を表し, ソフトウェア開発において保守作業に大きな影響を及ぼす可能性がある．なぜなら, コードクローン部分を修正する必要がある場合, コードクローンセット全体に修正を適用する必要がある．コードクローンに関する過去の研究論文や既存のコードクローン検出ツールは多数存在し, 今なおコードクローンに関する研究は活発に行われている [1][3]．本研究では, コードクローン検出ツールを活用し, 保守作業を支援する新たなツールを提案する．

2.2　コードクローン検出ツール

コードクローンを検出するツールはいくつかあるが，本研究では CCFinderSW を用いる．CCFinderSW は, ソースコード内のすべてのコードクローンを検出し, コードクローンセットの位置情報を出力するツールである．このツールは, 多言語プロジェクトに対応しており, 出力ファイルにはファイル情報とクローンセットの詳細な情報が記述されている．また, 言語の選択やクローンの最小サイズ, 特定のパスの除外など, さまざまなオプション機能が提供されており, ユーザーが独自の設定を行うことが可能である．CCFinderSW は, コードクローンに関する研究や，プロジェクト管理において非常に有用なツールである [2].

3　既存の検出ツールの課題

CCFinderSW は，コードクローンの正確な位置情報を提示してくれる．しかし，以下のような課題もある．

- コードクローンの位置は正確に教えてくれるが，実際にコードクローンを見つける際には，検出結果のファイルとソースコードを見ながら探す必要があり，手間が大きい.
- どのようなコードクローンなのかは，ソースコードを見なければわからない.
- ソースコードだけだと，変更を加えた部分がコードクローンに含まれているかが分からない.
- 大幅な変更を加えた際に，行数が変化してしまいコードクローンの位置が分からなくなる.

実際にコードクローンを確認するためには，自らの目でファイルパスを確認し，示された行まで移動しなければならない．ファイルパスを確認するのも，ファイルパス情報と，クローン ID の情報が膨大になっている場合，それぞれの情報が示されている部分が離れているため参照するのに手間がかかる.

また，ソースコードに変更を加えた際に，その部分にコードクローンが存在するのかどうかが分からない．よって，コードクローンと分からずに変更を加えてしまい，他のクローン片にも変更を加える必要があるにもかかわらず，変更をしていないという状況が起こり得る.

実際に確認しようとすると，CCFinderSW の出力ファイルから，修正を加えたファイルのファイル番号を探す．次に，そのファイル番号のかかわっているコードクローンを探し，その中に修正を加えた行数が含まれているかを確認しなければならない．さらに，修正を加えた部分がコードクローンに含まれていた場合，ファイル番号を確認し，ファイルパスを探し，指定された行を見てこちらにも修正を加える必要があるのかを検討できる．本研究ではこれらのコードクローンの管理に関する課題を解決し，作業の効率化をするツールを開発する.

3.1　既存ツールの問題点と提案する支援方法

既存の研究として佐々木ら [5] の研究がある．この研究では，CCFinder[4] に対して，コメントを利用し，コードクローン情報を付加している．しかし，佐々木らの開発したツールでは, 同じファイル内のクローン片はクローン ID を検索することにより，素早く見つける事ができるが，どのファイルにクローン片が含まれているのかが簡単には分からない.

さらに，1 行が複数のクローンセットに含まれている，

または重複している部分のあるクローンセットの場合，クローン ID が示されているコメント部分が長くなってしまい，可読性が低下してしまう．

本研究ではこれらの課題を解決し，作業の効率化をするコードクローン保守支援ツールを開発する．

本研究では，以下の機能を開発していく．

1. コードクローン情報付加ツール
   ソースコードにコードクローンに関する情報を付加する（ただし，編集するための一時的なファイルを作成し，元のファイルには付加しない）．
2. コードクローン情報を非表示にする機能
   コードクローン情報による可読性の低下を避けるため，一時的にコードクローンを非表示にする．
3. コードクローン部分にハイライトを付ける機能
   コードクローンがどこにあるのかが一目でわかるように，コードクローン部分にハイライトを付ける．
4. ファイルパス情報まで移動する機能
   クローン ID 部分を選択することで，ファイル下部に付加した，関連するファイルパス情報まで移動する．

1 の機能は Python，それ以外の機能は VScode の拡張機能を利用して開発する．

4 保守支援ツール

4.1 コードクローン情報付加ツール

ソースコードを見るだけでは，コードクローンかどうかが分からないという課題と，行数が変わってしまいコードクローンの位置が分からなくなるという課題に対し，ソースコードに直接コードクローン情報をコメントとして書き込むという方法で課題の解決を図る．

コードクローン情報付加ツールは，CCFinderSW の情報をソースコード内に付加する．これによりソースコードの編集時に，その部分がコードクローンであるかどうかを目視で判別できる．また，ソースコードの下部に関連するクローン片が存在するファイルパスも付加するため，CCFinderSW の出力結果のファイルを参照する必要がなくなり，コードクローンの見落としや，作業の効率化が期待できる．ただし，ここではソースコード中に直接書き込むのでなく，別のファイルとして出力する．

仮に作成した「calc.java」に対して, コードクローン情報付加ツールを実際に実行した例を以下に示す．変更前のソースコードを Listing1, 生成された変更後のソースコードを Listing2 に示す．

Listing2 で，コードクローンである行の後ろに，コメントとして, クローンＩＤを付加している．また, ソースコードの最終行の後に，このファイルに存在するコードクローンに関係しているファイルの情報を付加している．ただし，ここで付加しているクローンＩＤには，CCFinderSW の出力結果には存在していないが，クローンセットの要素にインスタンス番号を設けている．

Listing 1 変更前のソースコード（calc.java）

```
public class CodeCloneExample {
    public int addNumbers
    (int a, int b)
    {  return a + b; }
    public double rectangleArea
    (double len, double wid)
    {
        return len * wid;
    }
}
```

Listing 2 変更後のソースコード (output_calc.java)

```
public class CodeCloneExample {
    public int addNumbers  //CC.1,1
    (int a, int b) //CC.1,1
    {  return a + b; } //CC.1,1
    public double rectangleArea //CC.1,2
    (double len, double wid) //CC.1,2
    { return len * wid; } //CC.1,2
}
//Data.1,1,/home/username/
documents/calc.java
//Data.1,2,/home/username/
documents/calc.java
```

Listing1 を見ても，これだけではどこがコードクローンであるか分からない．しかし，Listing2 を見ると，2 行目から 4 行目と，5 行目から 7 行目がコードクローンであることが，ソースコードを見ただけで分かる．また，具体的にソースコード中にコメントとしてクローン ID を追加しているので，変更を加える際に行番号が変化することを気にせずに作業することができる．

さらに，ソースコードの下部に付加されたファイル情報によって，他のファイルに及ぶクローンセットでも，CCFinderSW の出力結果のファイルを参照する必要がなくなる．

4.2 コードクローン情報を非表示にする機能

ソースコード中に，コードクローン情報を書き込むことによって，複数のクローンセットに含まれている行の可読性が低下してしまうという課題に対して，コードクローン情報を非表示にするという方法で課題の解決を図る．

コードクローン情報を非表示にする機能は，第 3 章で説明した，コードクローン情報付加ツールによって付加された情報を非表示にする機能である．ソースコードの編集者が，コードクローン情報が不要で一時的に消したい場合に，

VScode 上で特定のコマンドを実行することで，コードクローン情報を一時的に非表示にすることができる．この機能により，可読性の低下が改善される．

実行例を以下に示す．前提として, 対象となるファイルには, 第 3 章に開発したコードクローン情報付加ツールによりコードクローンの場所とコードクローンの情報が付加されているものとする．

1. VScode では,「Ctrl+Shift+P」を同時に押すことでコマンドパレットを開くことができる．
2. コマンドパレットを開き,「Hide CC」というコマンドを入力し実行する．

この機能を実行すると（図 1）から（図 2）に変化する．２つの図を見てわかるようにコードクローンの横にある「//CC」のコメントが非表示になっていることがわかる．また、この非表示の状態は文字列を削除した状態ではないため, 同じファイルを再度開くと「//CC」がついた状態で開くことができる仕様となっている．

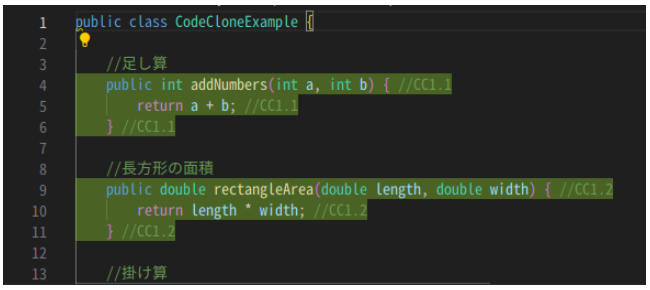

図 1　コマンド実行前

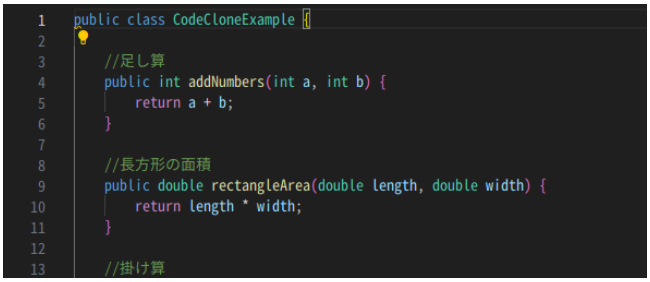

図 2　コマンド実行後

4.3　コードクローン部分にハイライトを付ける機能

コードクローン情報を非表示にした状態でも，コードクローン部分にハイライトを付けて表示することで，より可読性を向上させることができる．

コードクローンの詳しい情報がいらないが，ソースコードのどこにコードクローンが存在しているのか知りたい場合などに役に立つと考えられる．

この機能の実行手順を以下に示す．

1. 拡張機能をインストールする．
2. 本研究で開発したツールをかけたファイルを開く．

この機能はその他の機能と異なりコマンドが存在しないため, エディタにマッチする文字列がある場合に自動でハイライトが付く．通常は（図 3) のようにコードクローンの場所であってもハイライトが付いていないが, コメントが付加されたあとのソースコードでは（図 4) のようにハイライトが付く．

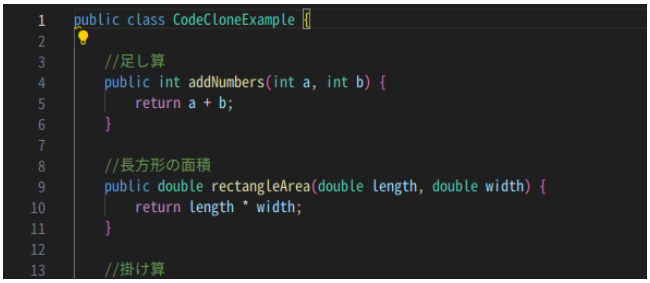

図 3　ハイライトが付いていないソースコード

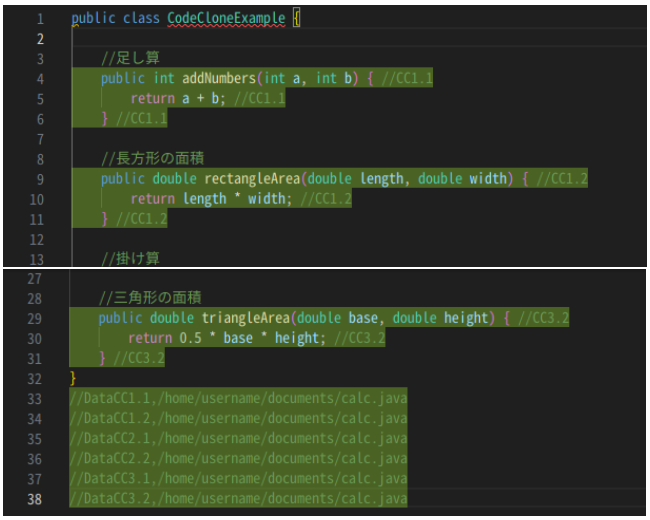

図 4　ハイライトが付いたソースコード

4.4　クローン ID を利用しパス情報まで移動する機能

3 章で説明した，コードクローン情報付加ツールにより他のファイルを参照しながら，コードクローンを探す必要はなくなったが，関連するファイルパス情報が，ソースコード下部にしかないため，膨大なソースコードである場合，画面をスクロールする必要がある．また，関連するファイルパス情報も多くあり，その中から関連するファイルパス情報を探すのは，手間である．なので，クローン ID を選択することで，そのクローン ID に関連するファイルパス情報が記されている場所にカーソルを移動する機能を作ることで，課題の解決を図る．

クローン ID を利用し，パス情報まで移動する機能は，関連するファイルパスの情報を知りたいクローン ID を選択したまま，特定のコマンドを実行することで，知りたい関連するファイルパス情報の位置にカーソルを移動する機能である．この機能により，より早く関連するファイルパ

ス情報を見つける事ができ，作業の効率化が期待できる．

この機能の実行手順を以下に示す．

1. 調べたいコードクローンの横にある「//CC..」をカーソルで選択する．
2. コマンドパレットを開く．
3.「Cursor CC」のコマンドを入力し，実行する．

(図 5) は 3 行目の //CC1.1 を選択した状態で「Cursor CC」のコマンドを実行すると 26 行目の //DataCC... にカーソルが移動したということを示している．

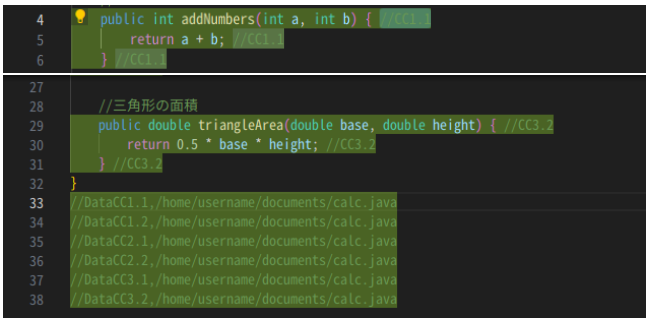

図 5　パス情報まで移動する機能

5　考察

本研究では，コードクローン情報付加ツールと VScode の拡張機能を利用した，コードクローン情報を非表示にする機能，コードクローン部分にハイライトを付ける機能，クローン ID を利用しパス情報まで移動する機能を開発した．コードクローン情報付加ツールによって，CCFinderSW の出力ファイルを参照する必要がなくなる．VScode の拡張機能を利用したコードクローン情報を非表示にする機能と，コードクローン部分にハイライトを付ける機能によって，一目でどこがコードクローンなのかが分かる．クローン ID を利用しパス情報まで移動する機能によって，ファイル情報を探す必要がなくなる．これらの機能を使うことで，コードクローンを保守，管理する際にとても効率的に作業することができる．

しかし，今回作成したコードクローン情報付加ツールを実行した後に，もう一度 CCFinderSW を実行すると，コードクローン情報付加ツールの出力ファイルも含まれてしまうため，正常に動作しない．従って，もう一度 CCFinderSW を実行する前に，コードクローン情報付加ツールの出力ファイルを消す必要がある．この問題の解決策として，既存研究で実装されていたコメント除去機能と，ファイルコピー機能が役に立つと考えられる．これは今後の課題である．

他にも，さらに効率化するための機能として，クローン ID をクリックするだけで次のインスタンス番号のクローン片に移動する機能や，コードクローン部分を抜き出し，Copilot を利用してコードクローンの内容を，1 つのファイルにまとめて出力したり，コメントとして付加したりする機能など，今後まだまだ発展，拡張していくことができると考える．また，本研究のツール，拡張機能は現在 Java にしか対応出来ておらず，CCFinderSW の対応している多言語に対応していく必要がある．

将来的には，ソースコードを見るだけでコードクローンの情報がすべて分かるようになる．また，必要な時だけ表示できる，クローン ID をクリックするだけで，クローンセットが存在するファイルを開くことができるツール，拡張機能にしていきたい．

しかし，本研究で作成したツール，拡張機能についての評価ができていない．本研究の目的が，コードクローンに関する保守作業の効率化であるため，複数のファイルにまたがるコードクローンを見つけるまでの移動時間の計測や，サンプルファイル上のソースコードで，実際にコードクローンの保守作業を行い，その時間の計測，フィードバックの収集などの方法で，評価していこうと考えている．

6　おわりに

本研究では，既存のコードクローン検出ツールの課題を解決するためのツール，拡張機能の開発を行った．開発したツール，拡張機能は，コードクローン情報付加ツールと VScode の拡張機能を利用し，コードクローン情報を非表示にする機能，コードクローン部分にハイライトを付ける機能，クローン ID を利用しパス情報まで移動する機能の 1 つのツールと 3 つの拡張機能を開発した．今後は，これらのツール，拡張機能の評価をすると共に，さらに利便性を高めるために，他の機能の開発など，発展させる必要がある．

参考文献

[1] 肥後芳樹, 楠本真二, 井上克郎:『コードクローン検出とその関連技術』電子情報通信学会 論文誌, 2008.

[2] 瀬村雄一，井上克郎：『多様なプログラミング言語に対応可能なコードクローン検出ツール CCFinderSW』，電子情報通信学会論文誌 D，Vol.J103-D，No.04，pp.215-227.

[3] Katsuro Inoue, Chanchal K. Roy, edited by: ”Code Clone Analysis: Research, Tools, and Practices”, Springer, ISBN 978-981-16-1926-7, Aug., 2021.

[4] T. Kamiya, S. Kusumoto and K. Inoue, ”CCFinder: a multilinguistic token-based code clone detection system for large scale source code,” in IEEE Transactions on Software Engineering, vol. 28, no. 7, pp. 654-670, July 2002, doi: 10.1109/TSE.2002.1019480.

[5] 佐々木 亨，肥後 芳樹，神谷 年洋，楠本 真二，井上克郎：『プログラム変更支援を目的としたコードクローン情報付加ツールの実装と評価』，電子情報通信学会論文誌 D-I J87-D-I (9), 868-870, 2004-09.