コードクローンの可視化と実装方法の検討
Visualization of Code Clones and the Implementation

安藤未緒† 前田和昭‡ 高橋道郎‡
Mio Ando† Kazuaki Maeda‡ Michio Takahashi‡
† 中部大学　経営情報学研究科　前期博士課程
‡ 中部大学　経営情報学部　経営情報学科
† Graduate School of Business Administration
and Information Science, Chubu Univ.
‡ Department of Business Administration
and Information Science, Chubu Univ.

要旨
ソフトウェア保守のためにコードクローン検出の研究が活発に行われてきた．従来のコードクローン検出手法では，検出結果を詳細に入手できるものの，何らかの規則を発見することは難しい．コードクローンは，ソースコード内部に散在する数多くのコード片間の複雑な関係となっているためである．そこで本稿では，コードクローン検出結果を無向グラフを使って可視化する方法を検討する．さらには，無向グラフを表示するための実装方法についても述べる．

1. はじめに

最近ではソフトウェアの規模が拡大するようになり，ソフトウェア保守が難しくなってきた．例えば，自動車に搭載するソフトウェアの場合，2000 年に 100 万行の規模だったものが，2009 年には 500 万行から 1000 万行まで拡大した[1]．この規模のソフトウェアを保守する作業の困難さは想像に難くない．

複雑な保守作業を支援するための一つの技術として，コードクローン検出があり，その研究開発が活発に行われてきた．コードクローンとは，ソースコード中に存在する類似したコード片のことである．コードクローンには明確な定義はないものの，文献[2]では，

「コードクローンとは，ソースコード中のある一部分（コード片）のうち，他のコード片と同一または類似しているものを指す．」

と述べられている．コードクローン検出ツールが違うと類似の判断が違ってくるため，検出結果にばらつきがある．

ソフトウェアが大規模になると，人手でコードクローンを探し問題を解決することは難しい．しかも，全てのコードクローンが問題となるわけではなく，また，コードクローンのタイプによって保守作業が違ってくる可能性もある．

従来のコードクローン検出手法では，検出結果を詳細に入手できるものの，そこからクローン間にある規則を発見することは難しい．ソースコード内部に散在するコードクローンの量が増え，それらが複雑な関係を作り出すため，検出結果をテキストで表示するだけでは，コードクローンの分布やコードクローン間の関係を把握することはできない．そこで，コードクローンを検出するだけではなく，コードクローンの分布状況を把握するために可視化ツールを使ってシステム保守を支援することが重要になってくる．本稿では，コードクローン検出結果を可視化する手法について検討し，その可能性について議論する．

2. 既存のコードクローン可視化

日本発のコードクローン検出ツールとして，CCFinderX がある[3]．CCFinderX を使えば，Java，C#，C/C++，COBOL，VB で記述されたソースコードからコードクローンを検出できる．CCFinderX は検出結果を表示するために散布図を使う．この散布図では，縦軸・横軸でファイル上の位置を表現してコー

ドクローンをプロットすることで，全体の分布を把握できる．例えば，図 1 に示すように，三つのファイルに跨ったコードクローンが存在したとする．CCFinderX がこれらのファイルを解析すると，図 2 に示すような散布図を作成する．1 対 1 の関係を表現するため，1-2，1-3，2-3 の 3 点をプロットし，さらには縦軸と横軸が同じスケールであることから，散布図の右上側と左下側が対称となるようにする．なお本稿では，図 1 のような同一のコードクローンの集合をクローンセットと呼ぶ．

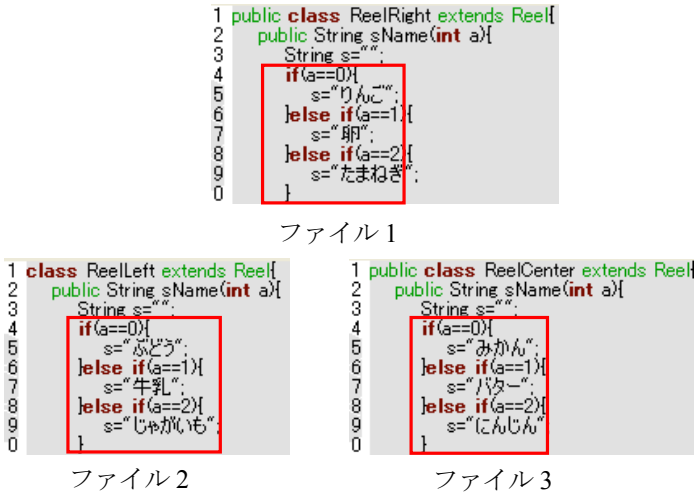

図 1　三つのファイルに跨るコードクローン

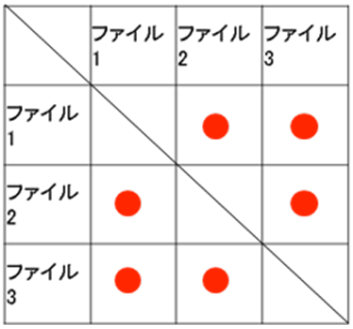

図 2　散布図の概略

コードクローンの検出結果を，車輪を使って視覚化した Atomiq が商用でリリースされている[4]．その実行例の一部を図 3 に示す(印刷の都合により背景色を変更済み)．ファイルを順番に輪の上に配置し，その上でコードクローン間の関係を表現している．ファイル内のコードクローンは隣り合う部分を曲線でつなぐため突起物のように表現される．ファイル間のコードクローンはアーチ状に表現される．この図があれば，ファイル内に存在するコードクローンと，ファイル間に存在するコードクローンを一望に把握することができる．しかし，1 対 1 の関係で線を引いているため，一つのコードクローンが複数のファイルに跨っている場合，その事実を直感的に把握することができない．

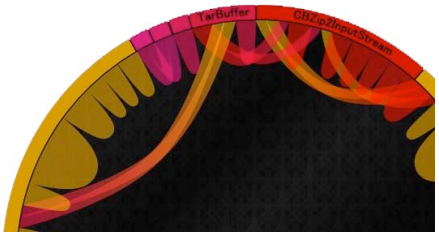

図 3　Atomiq の表示例

3. 散布図との比較

コードクローン検出ツールは，一般的にコードクローンを検出する部分と検出結果をわかりやすく可視化する部分に分かれている．表 1 で示すように CCFinderX では散布図を用いて可視化する．散布図は，ソースコード全体のコードクローンの分布の把握及び同一ファイル，または，二つのファイル間に存在するコードクローンを見つけることに向いている．しかし，複数のファイル間に跨って存在するコードクローンを見つけることが難しい．そこで，散布図ではできない，複数のファイル間に跨るコードクローンを把握するために，新しい可視化手法である FC グラフを考えた．本来は，CCFinderX の検出結果を用いて可視化を行うべきだが，検出結果のみを取り出すことができないため，他の検出ツール（CPD[5]）を使用し可視化部分の作成を行った．

散布図は，

・　長所：ソースコード中のコードクローン分布が把握可能
・　短所：複数のファイルに共通するコードクローンの発見が困難
という特徴を持つ．これに対して，FC グラフは，
・　長所：複数のファイル間に跨って存在しているコードクローンを発見可能
・　短所：同一ファイル内に存在しているコードクローンを非表示
という特徴を持つ．

表 1　検出ツールと可視化手法

	既存ツール		自作ツール
コードクローン検出	CCFinderX	CPD	CPD
可視化手法	散布図	テキストのみ表示	FC グラフ

4. FC グラフ

文献[6]では，コードクローンが存在するファイル群に着目して，同じファイル群に存在するコードクローン群をグループ化することで，コードクローン同士の関係をグラフで表現している．グラフの作成手順は，検出されたコードクローンのクローンセットを頂点とし，同じファイルにコードクローンが存在するクローンセット同士を線で結び，どの頂点とも結ばれなかった頂点を削除する．図 4 に同じファイル内にクローンセット 1，2，3 が存在している例を示す．

FC グラフは，保守を行うに当たって重要となるファイル間に跨って存在するコードクローンの可視化を目的とする．FC グラフは文献[6]のグラフとは異なり，ファイル，クローンセット，パッケージの三つを頂点とする．図 5 にファイル 1 と 3 がパッケージ abc に存在し，ファイル 1，2，3 にクローンセット 1 が存在している例を示す．

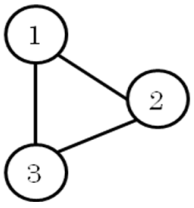

図 4　グラフの表示例

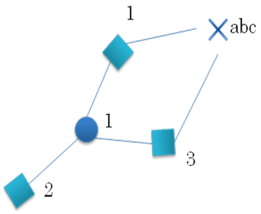

図 5　FC グラフの表示例

図 6 は実際に作成した FC グラフの一部分である．このグラフは四角がファイル，円がクローンセット，×印がパッケージを意味している．ファイルとクローンセットには零から順に振られた番号が，パッケージにはパッケージ名が表示される．また，クローンセットが存在しないファイルおよび，ファイルが一つしかないパッケージは表示されない．画像の丸で囲まれた部分は，クローンセット 241 がファイル 116,117,118,119,120 の五つのファイル間に存在することを意味している．

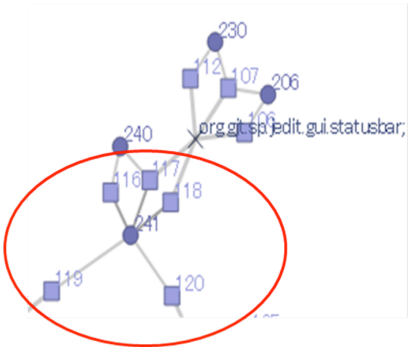

図 6　FC グラフの一部

CPD は，コードクローン検出結果を XML ファイルで出力できる．その XML ファイルを入力とし，Protovis[7]を使って Web ブラウザに表示可能な JavaScript コードを生成するためのプログラムを Java 言語で作成した．図 6 に示した FC グラフの一部は，Web ブラウザで表示された部分を切り取ったものである．

5. 可視化手法の比較

図 7 のように，三つのファイルにそれぞれ二つのクローンセットが存在する場合，散布図と FC グラフでは図 8 と図 9 のようになる．散布図は全体のコードクローンの分布状況を把握することに向いているが，複数のファイル間に跨って存在するコードクローン把握が難しい．FC グラフは同一ファイル内に存在しているコードクローンは表示されないが，複数のファイル間に跨って存在するコードクローンの発見が容易である．

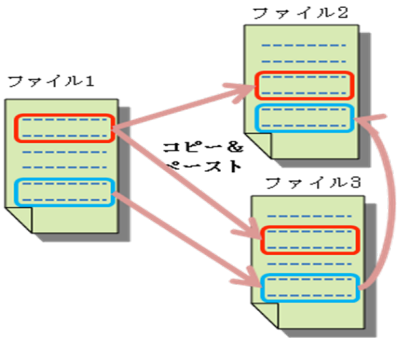

図 7　コードクローン例

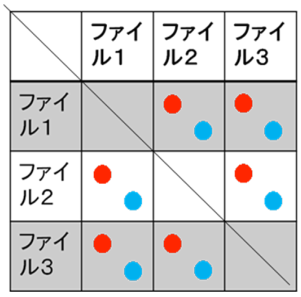

図 8　散布図

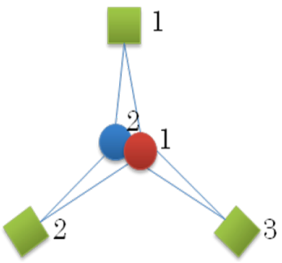

図 9　グラフ

6. まとめ

ソフトウェア規模の拡大・複雑化は保守にかかる労力及び費用の増加の原因となっている．このような，ソフトウェアを複雑にし，保守を妨げる要因の一つであるファイル間に跨るコードクローンを把握することが重要と考えた．既存のコードクローン検出ツールを使用したところ，可視化手法である散布図では，ファイル間に跨るコードクローンの把握に難があり，FC グラフを開発した．FC グラフを用いることにより，ファイル間のコードクローンを効果的に把握することが可能になった．今後，FC グラフを既存のソースコードに適用し，効果を検証する．

参考文献

[1] JASPAR, 自動車向け共通基盤ソフトウェア開発事業の成果発表会, 2010, http://www.meti.go.jp/press/20100129002/20100129002-3.pdf　(accessed at Nov. 5, 2010).
[2] 肥後芳樹, 吉田則裕, 楠本真二, 井上克郎, “産学連携の基づいたコードクローン可視化手法の改良と実装”, 情報処理学会論文誌, Vol. 48, No.2, 2007, pp.811-822.
[3] CCFinder ホームページ, http://www.ccfinder.net/ccfinderxos-j.html (accessed at Nov. 5, 2010).
[4] Atomiq : Code Similarity Finder, http://getatomiq.com/ (accessed at Nov. 5, 2010). .
[5] PMD - Finding Copied and Pasted Code, http://pmd.sourceforge.net/ (accessed at Nov. 5, 2010).
[6] 福島義彦, システム内に散在するコードクローンに着目したソフトウェア保守性向上の試み, 奈良先端科学技術大学院大学 情報科学研究科情報システム学専攻 修士論文, 2010.
[7] Protovis, http://vis.stanford.edu/protovis/ (accessed at Nov. 5, 2010).