ソフトウェア保守支援を目的とした コードクローン可視化ツール: Gemini

肥後　芳樹†　　楠本　真二†　　井上　克郎†

†大阪大学　大学院情報科学研究科　〒560-8531 豊中市待兼山町 1-3

E-mail:　†{y-higo, kusumoto, inoue}@ist.osaka-u.ac.jp

あらまし　ソフトウェアの保守作業を困難にしている原因としてコードクローンが挙げられる．コードクローンとは，ソースコード中の同一，または類似した部分を表す．あるコード片にバグが含まれていた場合，そのコード片のコードクローン全てについて修正の是非を検討する必要がある．我々の研究グループでは，コードクローンに対する保守支援を行うため，分析ツール Gemini を開発してきている．本稿では，Gemini の機能について述べる．

キーワード　ソフトウェア保守，コードクローン

1. 原稿用紙

ソフトウェアの保守作業を困難にする要因の 1 つとしてコードクローンが指摘されている[1]. コードクローンとは，ソースコード中に含まれる同一または類似したコード片のことであり，「重複コード」とも呼ばれる．コードクローンがソフトウェア中に作りこまれる原因として，既存コードのコピーとペーストによる再利用，頻繁に用いられる定型処理，パフォーマンス改善のための意図的な繰り返し，コード生成ツールによって生成されたコードなどがある[2]. コードクローンの存在が保守作業を困難にするのは，修正されるコード片のコードクローンが存在すれば，その全てのコードクローンに対して修正の是非を検討する必要があるからである．我々の研究グループでは，これまでにコードクローン検出ツール CCFinder[2]と分析ツール Gemini[4]を開発してきている．CCFinder は大規模なソフトウェアから実用的な時間でコードクローンを検出することが可能である. CCFinder の出力はテキストベースであり，コードクローンの位置情報はファイル名・行番号・列番号で表される．しかし，実システムに対してコードクローン検出を行うと，何万何十万というコードクローンが検出されてしまい，テキスト情報のみでは検出されたコードクローンの把握，理解は難しい．そこで，検出したコードクローンの分析には Gemini を用いる．Gemini はクローン散布図やメトリクスグラフなどさまざまなビューを用いてコードクローン情報の可視化を行う. Gemini を用いることで，ユーザは検出されたコードクローンの量や分布状態をより容易に把握することができると期待される．本稿では，Gemini を簡単に紹介する．

2. コードクローン

2.1. コードクローンの定義

あるトークン列中に存在する 2 つの部分トークン列 α，β が等価であるとき，α と β は互いにクローンであるという．またペア（α，β）をクローンペアと呼ぶ．α，β それぞれを真に包含する如何なるトークン列も等価でないとき，α，β を極大クローンと呼ぶ．また，クローンの同値類をクローンセットと呼ぶ．ソースコード中でのクローンを特にコードクローンという．

2.2. コードクローン検出ツール： CCFinder

CCFinder[2]}はプログラムのソースコード中に存在する極大クローンを検出し，その位置をクローンペアのリストとして出力する．検出されるコードクローンの最小トークン数はユーザが前もって設定できる．

CCFinder のコードクローン検出手順（ソースコードを読み込んで，クローンペア情報を出力する）は以下の 4 つの STEP からなる．

1. 字句解析：ソースファイルを字句解析することによりトークン列に変換する．入力ファイルが複数の場合には，個々のファイルから得られたトークン列を連結し，単一のトークン列を生成する．
2. 変換処理：実用上意味を持たないコードクローンを取り除くこと，及び，些細な表現上の違いを吸収することを目的とした変換ルールによりトークン列を変換する．例えば，この変換により変数名は同一のトークンに置換されるので，変数名が付け替えられたコード片もコードクローンであると判定することができる．
3. 検出処理：トークン列の中から指定された長さ以上一致している部分をクローンペアとして全て検出する．
4. 出力整形処理：検出されたクローンペアについて，ソースコード上での位置情報を出力する．

3. コードクローン分析ツール：Gemini

本節では, コードクローン分析ツール Gemini[4]の紹介を行う．全ての機能を紹介することが紙面の都合上

無理があるので，クローン散布図，メトリクスグラフについてのみ述べる．

3.1. クローン散布図

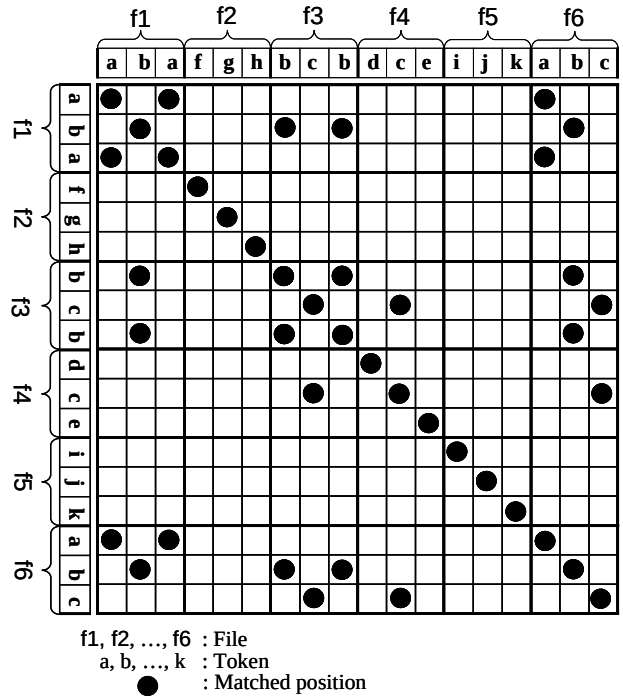

図1 クローン散布図モデル

クローン散布図の簡単なモデルを図 1 に示す．散布図の原点は左上隅にあり，水平軸，垂直軸は，それぞれソースファイルの並び(f1 から f6)に対応している．両軸上で，原点から順にそれぞれのソースファイルに含まれるトークンが並んでいる．座標平面内に点がプロットされている部分は，その両軸の対応するトークンが一致することを意味する．したがって，散布図の主対角線は，両軸の同じ位置のトークンを比較することになり，全ての点がプロットされることになる．一定の長さ(CCFinder で設定される最小一致トークン数) 以上の対角線分が，検出されたクローンペアである．図 1 では，f1 から f6 までのファイルが，それぞれ 3 つのトークンを含んでいる．ファイルの並びはファイル名の辞書順となっている．検出されるクローンペアは f1 と f6 に含まれるトークン列"ab"と，f3 と f6 に含まれるトークン列"bc"である．

3.2. メトリクスグラフ

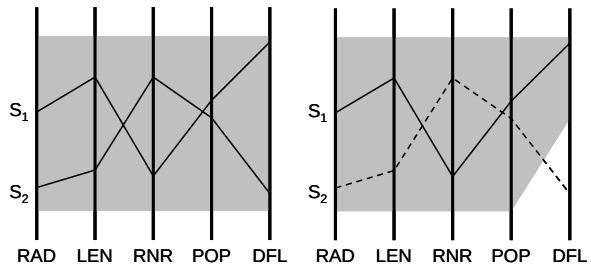

図2 メトリクスグラフモデル

メトリクスグラフは検出されたコードクローンを，メトリクスを用いて特徴づける．特徴づけされたコードクローンはクローンセット単位で表される．メトリクスグラフの簡単なモデルを図 2 に表す．このグラフで用いられているメトリクスは，RAD(S)，LEN(S)，RNR(S)，POP(S)，DFL(S)の 5 つである． ここでは，紙面の都合上 RNR(S)を除く 4 つのメトリクスについて簡単に説明を行う．

- RAD(S) : クローンセット S 内のコード片がどの程度ファイルシステム中に広く分散しているかを表す．例えば，全てのコード片が 1 つのファイル内に含まれる場合は 0，1 つのディレクトリ内に含まれる場合は 1 と，広く分散しているほど，この値は大きくなる．
- LEN(S) : S 内に含まれるコード片のトークン数の平均値を表す．
- POP(S) : S 内のコード片単位の要素数である．POP(S) が高いということは，同形のコード片が多く存在することになる．
- DFL(S) : S に含まれるコード片に共通するロジックを実装するサブルーチンを作り，各コード片をそのサブルーチンの呼び出しに置き換えた場合の減少が予測されるトークン数を表す．

5 つのメトリクスは図 2 の縦軸となっており，それぞれにメトリクス名のラベルがついている．このグラフではクローンセット毎に 5 つの縦軸上の点を結ぶ折れ線が引かれている．ユーザはこれら 5 つの座標の上限と下限を変更することで任意のクローンセットを選択することが可能である．例として図 2(右)は，DFL 軸の下限値を変更した状態を表している．この変更によって，図 2(左)では選択状態であったクローンセット S_2 が非選択状態となっている．

4. おわりに

本稿では，コードクローン分析ツール Gemini の機能を簡単に紹介した．今後は Gemini を用いて実際に用いられているソフトウェアの分析を進めていく予定である．

文　献

[1] M. Fowler, Refactoring: improving the design of existing code, Addison Wesley, 1999.

[2] T. Kamiya, S. Kusumoto, and K. Inoue, CCFinder: A multi-linguistic token-based code clone detection system for large scale source code, IEEE Transactions on Software Engineering, Vol.28, no.7, pp.654-670, Jul. 2002.

[3] 神谷年洋，"コードクローンとは，コードクローンが引き起こす問題，その対策の現状"，電子情報通信学会誌 Vol.87，No.9，791-797，Sep. 2004.

[4] 植田泰士，神谷年洋，楠本真二，井上克郎，"開発保守支援を目的としたコードクローン分析環境"，電子情報通信学会論文誌，Vol.86-D-I, No.12, pp.863-871, Dec. 2003.