ソースコードの文字数を利用したコードクローン抽出技術

安藤　優作[†1]　　垣谷　広輝[†1]　　平山　雅之[†1]　　菊地　奈穂美[†2]

近年のシステム開発において，プログラムの保守性を低下させる原因の一つとしてコードクローンが挙げられる．保守性を向上するためにはコードクローンを早期に検出し，除去する必要がある．本研究では早期のコードクローン検出のために，未完成（コンパイル完了前）コード内のコードクローンを行数や文字数といった情報を用いて，形状から効率的に検出する方法を提案する．

1.　はじめに

近年，システム開発は既存資産に機能を追加する方式が主流である．こうした開発ではソースコードの部分的コピーやペーストにより，コードクローンが作られることが問題視されている．特にコードクローンは修正コスト等を考えると開発の初期段階で確認し，除去する方が効率的である．このため，本研究ではコードクローンを早期に検出する方法を提案する．

コードクローンとは，ソースコード内で既存のコードを引用して作成された類似，または一致する処理が記述されたコード部分のことをいう．コードクローン検出の既存ツール[1][2]の多くでは，構文解析を用いて検出を行っている．この方法の場合，対象となるソースコードはプログラム言語の文法に従っていることが必要であり，開発初期の未完成（コンパイル完了前）コードでのクローンの検出は難しい．これに対し，本研究では未完成コードのコードクローン検出を目的として，コードブロックの形状からコードクローンを効率的に検出する方法を提案する．

2.　提案手法

提案手法では，ソースコードを構成するコードブロックに着目し，その行数や文字数といった形状特徴を利用して，以下のStep1~Step3よりコードクローンを検出する．

Step1:ソースコードを構成するブロックに分け

Step2:各ブロックの形状を比較し，クローンの候補ブロックを抽出する．

Step3:上記のクローン候補ブロックを精査し，クローン確率が高いブロックを特定する．

Step1 ソースコードのブロック分け

Step1 では，クローンの検出を効率的に行うために予めソースコードを構成している処理の塊をコードブロックとして検出する．

ソースコードを構成するブロックに分ける方法として，（A）：改行に着目した方法．（B）:制御文に着目した方法，の2つを併用する．なお，2行以下のブロックはprintf 文やプロトタイプ宣言により作られたブロックである可能性が高いため，ブロック分けは行うがクローンブロックの候補とはしないものとする．

（A）改行によるブロック分割

ソースコードでは多くの場合，構成するブロックは改行文字により分けられている．そのため，改行文字を用いてブロックとして分割する方法を採用する．具体的な方法としては，ソースコードの中で改行を見つけたらそこをブロックの区切りとする．これをコードの終わりまで続けることで，一つのコードを複数のブロックの塊に分割する．図1が実際に改行を境にブロック分けしたものとなる．

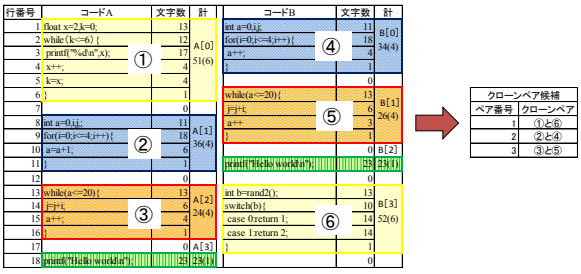

図1　ブロック抽出(改行)

（B）制御文によるブロック分割

上記（A）の手法で採用した改行に着目したブロック分割手法は改行がなければ適用できない．改行を用いたブロック分割が行われない場合には繰り返し文や分岐文などの制御文を認識し，ブロック分割する方法を提案する．制御文の書き方はプログラム言語によって若干の差異がある．ここでは，C やJava で書かれたプログラムを念頭に置き，それらの制御文では必ず” {”, ”} ”の2つがペアとして使用されていることを利用する．よって，” {”, ”} ”をブロックの区切りとして検出し，ブロック分割を行う．実際にブロック分割したものが図2となる．

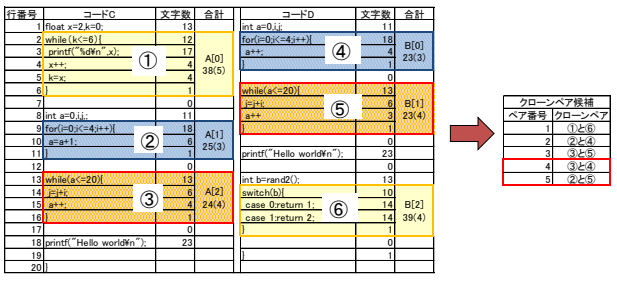

図2　ブロック抽出(制御文)

†1 日本大学理工学部
†2 沖電気工業株式会社

Step2　クローン候補ブロックの抽出

Step1 で抜き出された各ブロックについて，ブロック内の総文字数，行数を用いて形状評価を行い，各ブロック同士での類似度を比較する．これにより，クローンの候補となるブロックを形式的に特定する．類似度として，二つのブロックを比較した際の総文字数．行数の差を求める．この差が総文字数±30%，行数±10%以内に収まっている場合は二つのブロックは形状が似ていると判定する．

例えば，図 1 の左下の③ブロックでは総文字数 24 文字，行数 4 行という情報があることになる．これに対して，コード B を構成する④-⑥の各ブロックと比べると⑤ブロックが総文字数 26 文字，行数 4 行であり，行数±30%，総文字数±10%の差に収まっており，③と⑤がクローン候補のペアとして（ペア番号 3）抽出できる．このようにして，図 1 に示すようにクローン候補のペアが検出できる．

一方で，図 2 は制御文に着目したブロック分割を例示したものであるが，クローン候補のペア番号 4 と 5 は図 1 の改行を使用した手法では検出されなかったクローン候補である．

Step3　クローンブロック候補の精査

Step2 により抽出されたクローンブロック候補を対象に，より詳細な特徴の比較を行うことでクローンブロックの特定を行う．ここでは，詳細な特徴比較の方法としてブロック構成行ごとの文字数を評価し，クローンブロック候補の特定を行う．

Step2 でクローンである可能性が高いとして抽出されたブロックを対象にして，1 行あたりの文字数をそれぞれ比較する．これにより，Step2 で抽出されたものの中で構成が全く違っているにも関わらず，偶然一致してしまったものをクローンブロックとしないようにすることができる．図 3 では 1 行につき配列を 1 つ用意し，その配列番号が同じもの同士を比較する．例えば A[0]と B[0]，A[1]と B[1] といった形で中身を比較する． 全配列の一致率を平均して±60%を超えた場合はクローンである可能性が非常に高い組として検出する．検出した行は図 3(a)に示すように，一致したものに色を付けて表している．今回使用した例ではコード B の方で a++の後にセミコロンがないため，コードとしては誤ったものとなっており，文字数も変化しているが，他の行に対する配列の比較は一致している．そのため，変更されたクローンである確率が高いと認識できる．

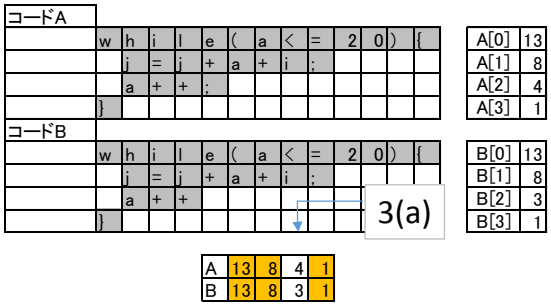

図 3　1 行ごとの文字数比較(行数一致)

ここで，ブロックの行数が違っていた場合には同じ行番号同士で比較するパターンだけでなく，行をずらした形でのパターンの比較も行う．その中で一致率が 60%以上のパターンが見つかれば，その二つのブロックをクローンブロックであると判定する．図4に行数が違う場合の1行当たりの文字数比較を行った例を示す．4(a)が比較結果である．この場合はどちらのパターンも一致率が 60%未満のため，クローンではないと判定する．

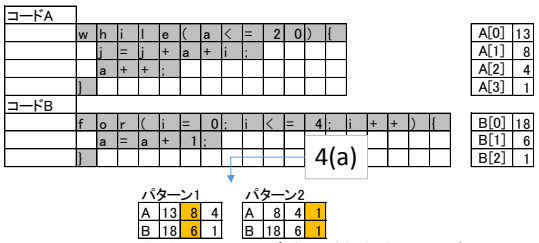

図 4　1 行当たりの文字数比較(行数不一致)

3.　一致率の算出

Step3 で用いた一致率の計算についてここで示しておく．図 5 に，図 3 で使用した文字数比較の一致率算出を示す．行番号ごとにそれぞれの文字数差と許容誤差を比較し，比較対象がその範囲内であれば一致とする．

ここで，対象となる両コードの一行当たりの文字数を，それぞれ平均した値の±10%を許容誤差とする．もし，コード A の文字数が 12，コード B の文字数が 18 であるなら，（12+18）÷2×0.1＝1.5 となり，四捨五入し，±2 文字を許容誤差とする．

実際に一致率を算出する例として，図 5 を示す．これは図 3 の比較を表として表したものである．図 5 の行番号 1 を見ると，コード A，B ともに文字数は 13 文字である．そのため，文字誤差は 0，許容誤差が±1 なので，行番号 1 は一致と判定される．

一致した行が全行番号に対してどの程度あるかを一致率として求める．図 5 の例ではブロック同士の一致率は 4 行中 3 行が一致しているため 75%となり，閾値とした±60%以内に収まっている．従って，この二つのブロックはクローンである確率が高いと判定される．また，図 4 のパターンに分けた形式でも考え方は同様に行い，一致率を算出する．

行番号	1	2	3	4
コードA	13	6	4	1
コードB	13	6	3	1
文字数差	0	0	1	0
許容誤差	±1	0	0	0
一致判定	○	○	×	○

$$\text{一致率}=\frac{\text{一致行数}}{\text{全行数}}=\frac{3}{4}=0.75=75\%$$

図 5　行数比較

4.　まとめ

本稿では，ソースコードを構成するブロック単位で文字数や行数などの形状情報や各ブロックの特徴を表す評価指標を用いてコードクローンを検出する方法を提案した．この方法は構文解析等を用いないため，文法的な誤りが含まれる開発途中のソースコード中でもコードクローンを検出することができる．今後，この方式を実装し実用性等の評価を進めていく予定である．

参考文献

[1] Lutz Prechelt, Guido Malpohl, Michael Philippsen “Finding Plagiarisms among a Set of Programs with JPlag” Journal of Universal Computer Science,1Vol.8,No.11/2002

[2]山中　裕樹，崔　恩瀞，吉田　則裕，井上　克郎，佐野　建樹，“コードクローン変更管理システムの開発と実プロジェクトへの適用”，情報処理学会論文誌，Vol.54,No.2,2013 年 2 月