

コードクローン検出とその関連技術

60206194 野口耕二郎

資料：電子情報通信学会論文誌 2008/6 Vol. J91-D No.6 pp.1465-1481
サーベイ論文 コードクローン検出とその関連技術

目次

- コードクロンの説明
- コードクロンの検出技術の紹介、比較
- コードクロンの応用
- まとめ

コードクローンの説明

コードクローンとは？

- ・ ソースコード中に存在する互いに一致または類似したコード片.
- ・ ソフトウェアの保守を困難にするといわれている.
コードクローン中にバグがあると、全て修正する必要がある.
- ・ 一方でコードクローンをそのまま用いるのが好ましい場合もある.

コードクローンの発生原因(一部)

- ・ コピー：再利用したい箇所をそのままコピーして持っていきたい場所に貼り付ける.
- ・ 定型処理：定義上簡単で頻繁に用いられる処理.
(例)キュー，給与税の計算
- ・ 処理系の機能の欠如：機能を補うために同じようなアルゴリズムを持つ処理を繰り返し書く.

コードクロンの定義

- ・ タイプ1：空白やタブの有無，括弧の位置などのコーディングスタイルを除いて完全に一致するコードクロンを指す.
- ・ タイプ2：変数名や関数名などのユーザ定義名，また変数の型などの一部の予約のみが異なるコードクロンを指す.
- ・ タイプ3：タイプ2における変更に加えて，文の挿入や削除，変更が行われたコードクロンを指す.

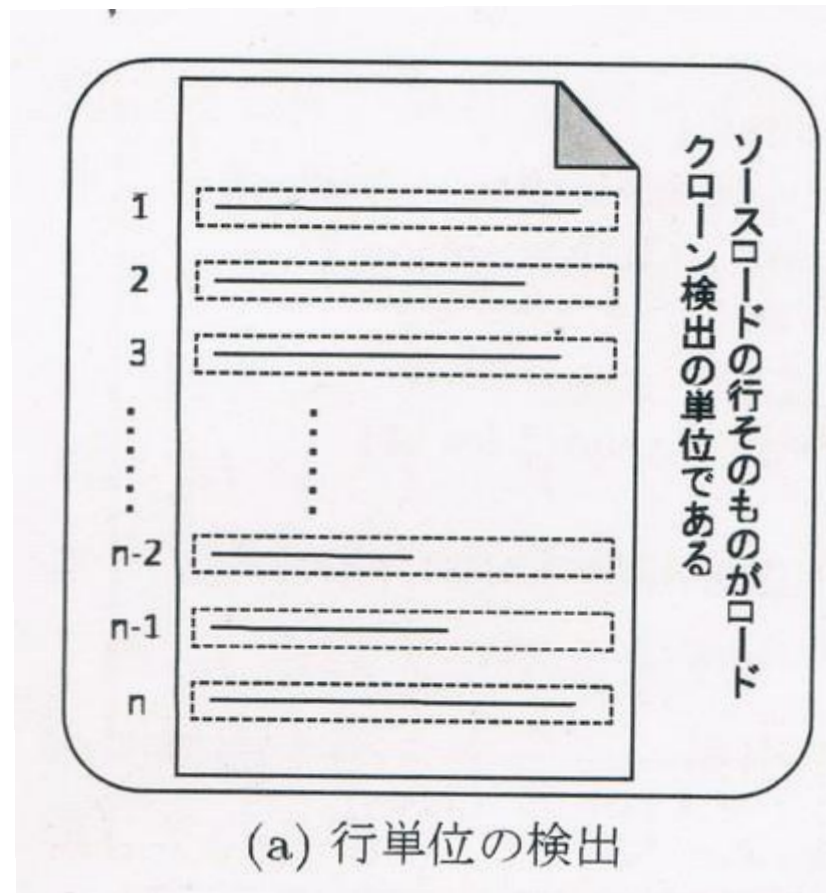
コードクローンの検出技術の紹介, 比較

コードクローン検出技術

検出技術の分類	検出手法	検出可能なコードクローン			対応言語
		タイプ1	タイプ2	タイプ3	
行単位の検出	Jonsonの手法	○	-	-	不特定多数
	Ducasseらの手法	○	-	-	不特定多数
	Bakerの手法	○	○	-	不明
	Wettelらの手法	○	-	○	不特定多数
字句単位の検出	Precheltらの手法	○	○	-	C/C++, Java, C#, Scheme
	Kamiyaらの手法	○	○	△	C/C++, Java, COBOLなど
	Liらの手法	○	○	○	C/C++
	Basitらの手法	○	○	○	Java
抽象構文木を用いた検出	Baxterらの手法	○	○	-	C/C++, Java, Ada, C#など
	Koschkeらの手法	○	○	-	C
	Jiangらの手法	○	○	○	C, Java
	Wahlerらの手法	○	○	-	XMLで表現されたソースコード
プログラム依存グラフを用いた検出	Kommondoorらの手法	○	○	○	C/C++, Scheme
	Krinkeの手法	○	○	○	C
メトリックスを用いた検出	Mayrandらの手法	○	○	○	C/C++
	Kontogiannisらの手法	○	○	○	C

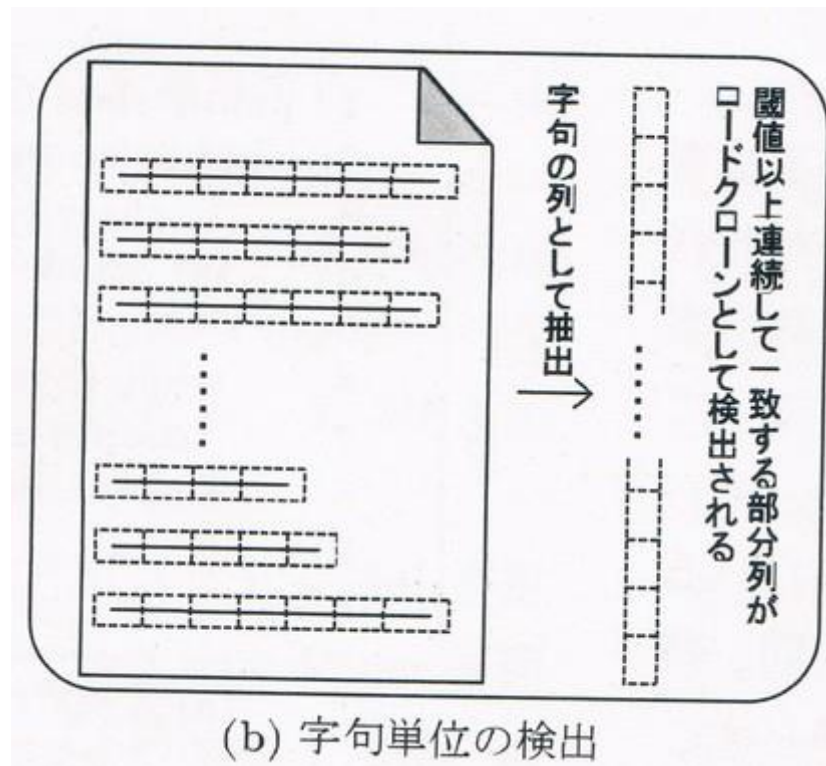
行単位の検出

- ソースコードを行単位で比較し、重複コードを特定する手法.



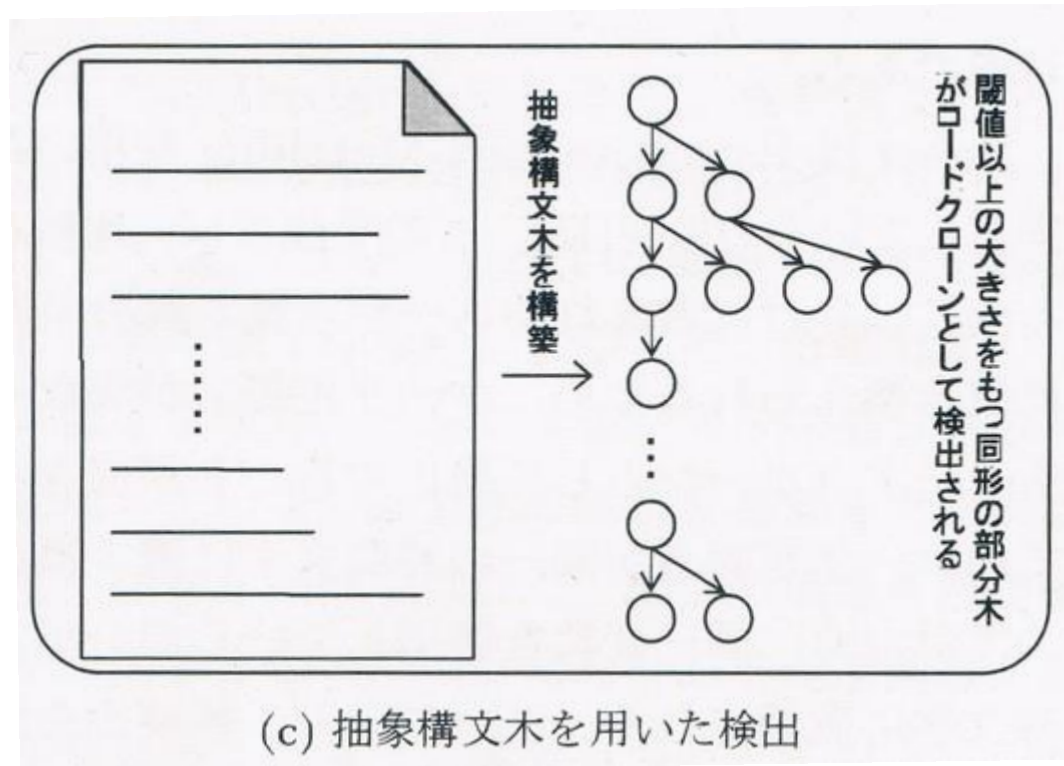
字句単位の検出

- 前処理としてソースコードを字句の列に変換。
しきい値以上連続して一致している字句の部分列をコードクローンとして検出する手法。



抽象構文木を用いた検出

- 前処理として構文解析を行い，抽象構文木を構築する．同型の部分木をコードクローンとして検出する手法．

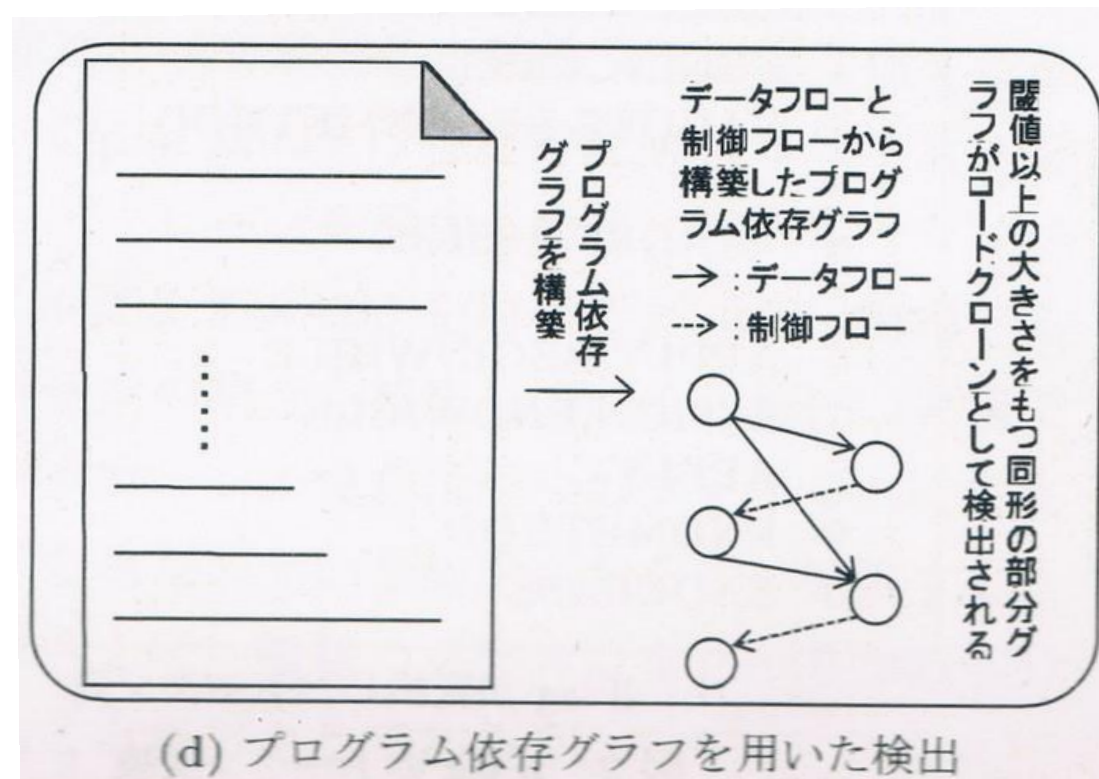


抽象構文木

- 構文木から言語の意味に関係のない情報を取り除いた木構造のデータ構造.
- コンパイラの過程の中で、意味解析が行われる前に生成される.

プログラム依存グラフを用いた手法

- 意味解析後，ソースコードの要素間の依存関係を表すグラフを構築し，同型のグラフをコードクローンとして検出する手法.



プログラム依存グラフを用いた手法

- 順序入れ替わりコードクローンや巻きつきコードクローンなど，他の手法では検出できないコードクローンを検出できる.

```
fp3 = lookaheadset + tokensetsize;
for (i = lookaheadset; i < k; i++){
bb  fp1 = LA + i * tokensetsize;
bb  fp2 = lookaheadset;
bb  while (fp2 < fp3)
bb      *fp2++ |= fp1++;
}
```

(a) コード片 1

```
fp3 = base + tokensetsize;
...
if (rp) {
    while ((j = *rp++) >= 0) {
        ...
        fp1 = lookaheadset;
        fp2 = LA + j * tokensetsize;
        while (fp1 < fp3)
            *fp1++ |= *fp2++;
    }
}
```

(b) コード片 2

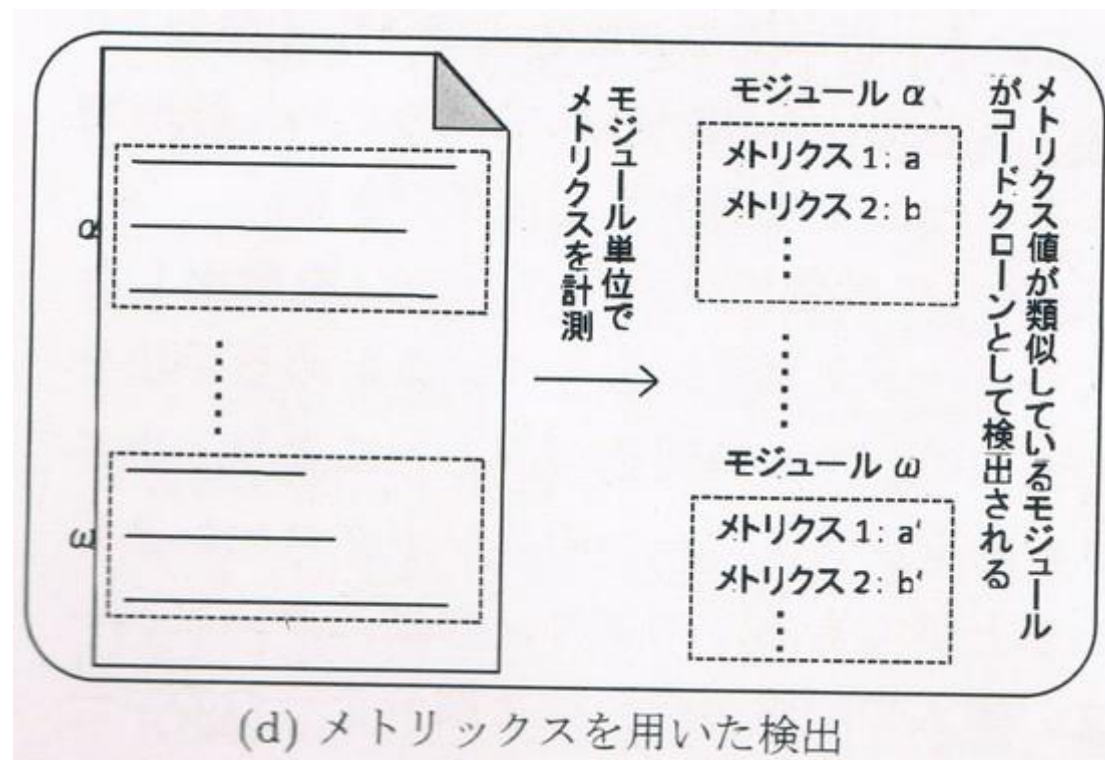
図 3 順序入れ替わりコードクローン
Fig. 3 Reordered code clone.

```
...
bb  tmpa = UCHAR(*a);
##  tmpb = UCHAR(*b);
bb  while (blanks[tmpa])
bb      tmpa = UCHAR(*++a);
##  while (blanks[tmpb])
##      tmpb = UCHAR(*++b);
bb  if (tmpa == '-') {
        tmpa = UCHAR(*++a);
        ...
    }
##  else if (tmpb == '-') {
        if (...UCHAR(*++b)...) ...
    }
```

図 4 巻き付きコードクローン
Fig. 4 Intertwined code clone.

メトリックスを用いた手法

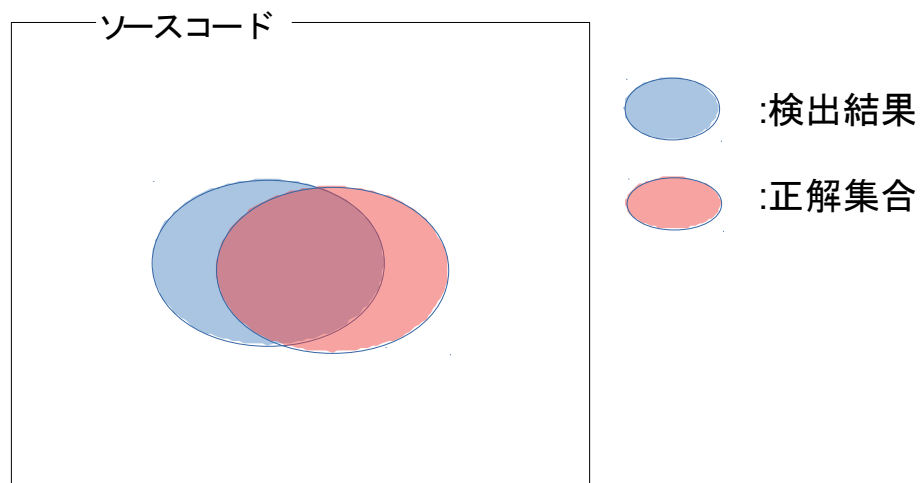
- プログラムのモジュールに対してメトリックスを用い、値の一致や近似の度合いでコードクローンを検出する手法.



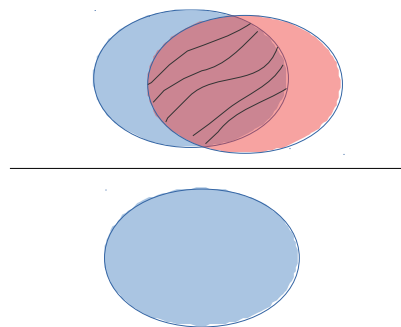
手法の比較

- Bellonらによる比較
- Bakerの手法, Ducasseらの手法, Kamiyaらの手法, Baxterらの手法, Krinkeの手法, Mayrandらの手法を比較.
- 約32万のコード片のペアのうち2%のソースを閲覧. コードクローンであると判断した正解集合を8つのOSSに埋め込み, 検出結果の適合率と再現率を比較.

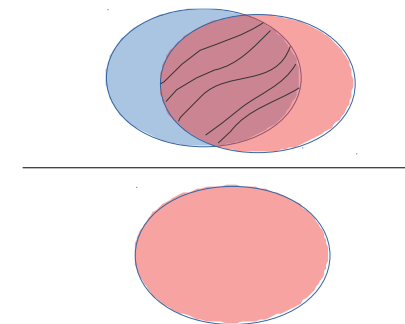
適合率と再現率



適合率 =



再現率 =



比較に用いる手法の説明

- Bakerらの手法（タイプ1, 2を検出）

比較を行う前に，各行に存在するユーザ定義名部分を特殊文字に置換し，比較して検出．

- Ducasseらの手法（タイプ1を検出）

各行に含まれる空白やタブが取り除かれた後，全ての行を比較して重複した行を検出．

比較に用いる手法の説明

- Kamiyaらの手法（タイプ1, 2を検出）

ユーザ定義名を特殊文字に変換し，接尾辞木検索アルゴリズムを用いて検出する手法.

- Baxterらの手法（タイプ1, 2を検出）

各部分木からハッシュ値を生成し，ハッシュ値が同一の部分木の対をコードクローンとして検出する手法.

比較に用いる手法の説明

- Krinkeの手法（タイプ1, 2, 3を検出）

字句をノードとするプログラム依存グラフを作成し, そのグラフからコードクローンを検出する手法.

- Mayrandらの手法（タイプ1, 2, 3を検出）

関数に対して21種類のメトリックスを計測しコードクローンを検出する手法.

比較の結果

- Bakerの手法, Ducasseらの手法, Kamiyaらの手法は検出結果が類似しており, 再現率も高い.
- Baxterらの手法, Mayrandらの手法は適合率が高い.
- Krinkeの手法はタイプ3のコードクローン以外ではうまく機能していなかった.

比較の結果

- Baxterらの手法で24%, Krinkeの手法で77%の検出結果が棄却された.
- 埋め込まれたコードクロンの内, 24%~46%が検出された.
- 性能面では, Bakerの手法, Kamiyaらの手法, Mayrandらの手法は高速. Baxterらの手法, Krinkeの手法は計算コストが高い.

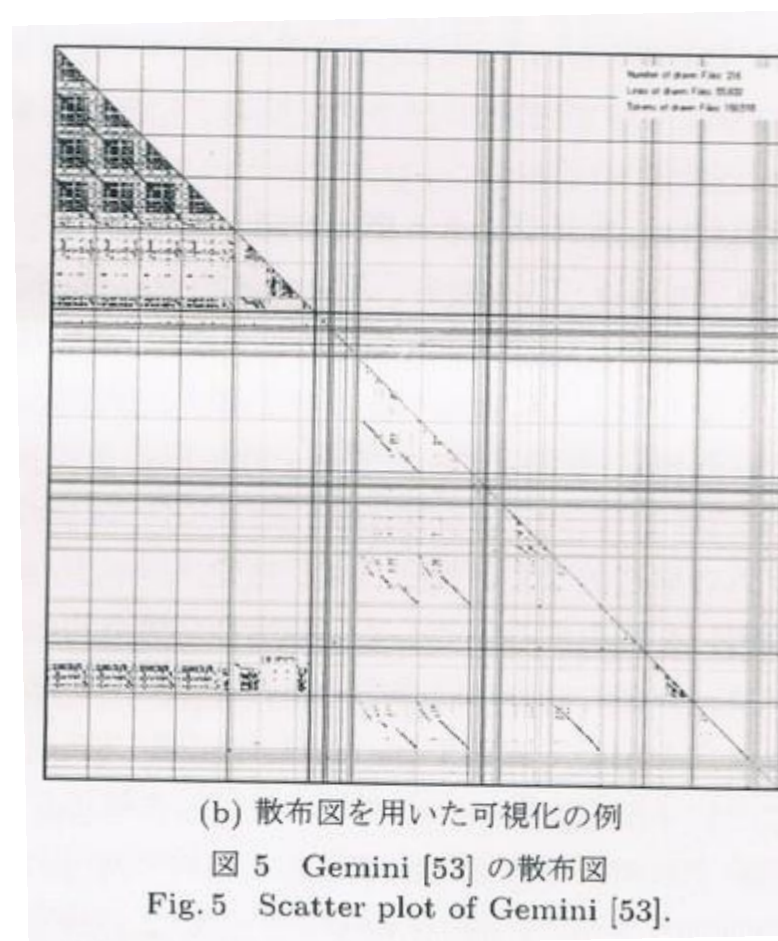
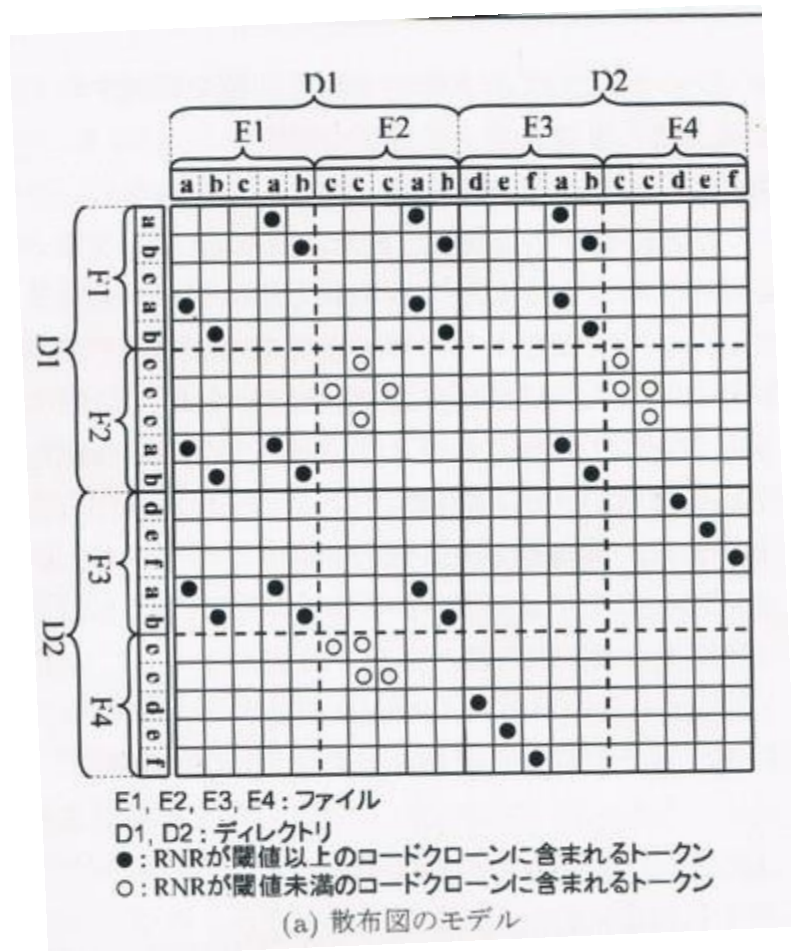
コードクローンの応用

コードクローンの分析

- 調査の必要のないコードクローンは調査の必要があるコードクローンを隠蔽し、分析の作業の非効率化を招く。
→肥後ら、非繰り返し度を示すメトリックス RNRを提案。しきい値として0.5が有効.

コードクローンの可視化

- Geminiの散布図



コードクローンの集約

- 互いにコードクローンになっているコード片群を1つのモジュールにまとめること.
- 集約することでコードクローンによって引き起こされる問題を軽減することが出来る.

コードクローンの修正支援

- 様々な理由でコードクローンの集約をすべきでない場合がある.
- あるコード片に対して修正を施すと, そのコードクローンに対して自動的に修正が施される技術がある.

まとめ

- コードクローンはバグの発生などでソフトウェア開発・保守の妨げとなることがある.
- コードクローンには3種のタイプがあり, どのタイプを検出できるかは検出方法によって異なる.
- コードクローンの検出方法は多くあり, 適材適所で用いる必要がある.