

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の適用

s161053 前田欣耶

A series of horizontal lines of varying lengths and colors (teal, light blue, and white) extending from the left edge of the slide towards the right, positioned below the author's name.



山中裕樹 院2回生



崔恩瀟 院2回生



吉田則裕
奈良先端科学技術大学院大
学情報科学研究科助教



井上克郎
大阪大学大学院情報科
学研究科研究科長



佐野建樹
日本電気株式会社ソフトウェア
生産革新部シニアエキスパート

目次

- 1、はじめに
- 2、コードクローン分類手法
- 3、コードクローン変更管理システム
- 4、適用実験
- 5、まとめ

1、はじめに

- コードクローンとは、ソースコード中に互いに一致または類似した部分を持つコード片。

```
341:    dwFrameGroupLength = 1;
342:    for(dwCnt = 2; dwCnt <= 64; dwCnt *= 2)
343:    {
344:        if(((ulOutRate / dwCnt) * dwCnt) !=
345:            ulOutRate)
346:        {
347:            dwFrameGroupLength *= 2;
348:        }
349:    }

784:    frameGroupLength = 1;
785:    for (cnt = 2; cnt <= 64; cnt *= 2) {
786:        if (((rate / cnt) * cnt) != rate)
787:            frameGroupLength *= 2;
788:    }
```

図1 変数名等が変更されているコードクローンの例

- コードクローンの存在はソフトウェアの保守を困難にする。

- ソースコードからコードクローンを検出するツール「CCFinder」とは...

表1 コードクローン検出ツールの比較

ツール名	検出方式 (粒度, マッチングアル ゴリズム, 計算複雑さ)	スケーラビリティ (プログラムテキストの規模, CPU, メモリ, 検出時間)	プログラミング言語への依存性 { 適用可能な言語 }
Dup [1] [2]	行, サフィックス木, $O(n)$	1.1MSLOC R3000 40MHz, 256MB RAM, 7分	小 {C}
Duploc [7]	行, 表検索, $O(n^2)$	460KSLOC G3 230MHz, 100MB RAM, 6.5 時間	小 {C, COBOL, Python, Smalltalk}
CCFinder [11]	字句, サフィックス木, $O(n)$	10MSLOC PIII 650MHz, 1GB RAM, 68分	中 {C, C++, Java, COBOL}
CloneDR [5]	AST の節点, 独自 (本文参照) $O(n^2)$	100KSLOC, (不明), 600MB RAM, 2 時間	大 {C, COBOL, Java}
SMC [3] [4]	メソッド ソーティング及び表検索 $O(n)$	215KSLOC, Pentium-Pro 180MHz, 64MB RAM, (不明)	大 {Java}

- ソースコードからトークン列(字句)として検出

- 変数名、メソッド名、空白の違いを無視

- ある閾値より長く一致したトークン列の範囲をコードクローンとして検出

• コードクローンの変更管理の必要性

- 同時修正...クローンセット中のコード片を一貫して修正。
 - 集約...クローンセット中のコード片に共通するロジックをサブルーチンにまとめる。
 - 位置情報の記録...ソースコード中にコメントとしてコードクローンとなっている他のコード片の位置を書く。
-
- コードクローンを人手で確認し、変更するのは非効率。システム(CCFinderに変更を加えた)を用いて、変更管理を行う。

2、コードクローンの分類手法

V_t

最新バージョン
のソースコード

V_{t-1}

最新からひとつ前のバー
ジョンのソースコード

C_i

V_i のソースコードに含まれる全ての
コードクローンの集合

• 本手法の手順

手順1 : V_{t-1}, V_t 全体にCCFinderを適用しコードクローン集合 C_{t-1}, C_t の検出を行う。

手順2 : コードクローンの変更履歴を分析するために C_{t-1} に含まれるコードクローンと C_t に含まれるコードクローンの対応関係を求める。
(・コードクローンの親子関係)

手順3 : 2バージョンのソースコードに含まれる全てのコードクローンを分類する。(・コードクローンの分類)

手順4 : 2バージョンのソースコードに含まれる全てのクローンセットを分類する。(・クローンセットの分類)

• コードクロンの親子関係

あるコードクロン $A \in Ct-1$ に対応するコードクロンが $B \in Ct$ である場合、コードクロン B をコードクロン A の子クロン、コードクロン A をコードクロン B の親クロン。

```
341:   dwFrameGroupLength = 1;
342:   for(dwCnt = 2; dwCnt <= 64; dwCnt *= 2)
343:   {
344:       if(((ulOutRate / dwCnt) * dwCnt) !=
345:           ulOutRate)
346:       {
347:           dwFrameGroupLength *= 2;
348:       }
349:   }
```

A

親クロン

```
784:   frameGroupLength = 1;
785:   for (cnt = 2; cnt <= 64; cnt *= 2) {
786:       if (((rate / cnt) * cnt) != rate)
787:           frameGroupLength *= 2;
788:   }
```

B

子クロン

図 1 変数名等が変更されているコードクロンの例

- コードクローンの親子関係の求め方。

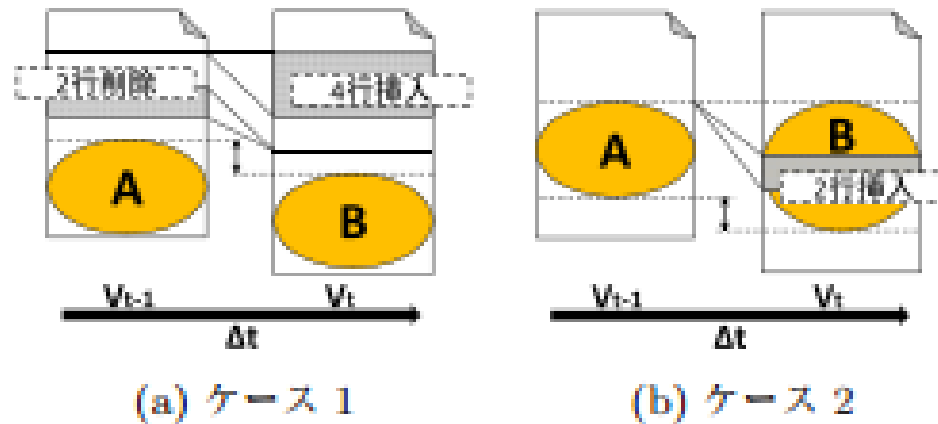


図 1 コードクローンの親子関係

Fig. 1 Parent-child relationship of code clones.

- (a)... 2 行削除。4 行挿入。Bの開始行番号、終了行番号はAのそれぞれに 2 足した値になる。
- (b)...AとBの開始行番号は一緒で、Bの終了行番号はAに 2 足した値になる。

このように対応するコード片を開始行番号、終了行番号によって求めて、BがCtに含まれるならAとBは親子となる。

• コードクローンの分類

Xをコードクローンとした時に...

$P(X)$

Xの親クローン
が存在する

$C(X)$

Xの子クローン
が存在する

$CP(X)$

Xの親クローンとXが
クローンペアである

$M(X)$

前バージョンからX
が編集されている

- Stable Clone : $P(X) \wedge \neg M(X)$ を充足するコードクローン $X \in Ct$ とその親のクローン。 **2バージョン間で変更がないコードクローン。**
- Modified Clone : $P(X) \wedge M(X) \wedge CP(X)$ を充足するコードクローン $X \in Ct$ とその親クローン。 **編集後も同じクローンセットに属するコードクローン。**
- Moved Clone : $P(X) \wedge M(X) \wedge \neg CP(X)$ を充足するコードクローン $X \in Ct$ とその親クローン。 **大幅な編集がなされたため、異なるクローンセットに属するようになったコードクローン。**
- Added Clone : $\neg P(X)$ を充足するコードクローン $X \in Ct$ 。コード片のコピーアンドペーストなどによって **Vt で新たに発生したコードクローン。**
- Deleted Clone : $\neg C(X)$ を充足するコードクローン $X \in Ct-1$ 。クローンセットの集約やコード片の削除などによって **除去されたコードクローン。**

・ クローンセットの分類

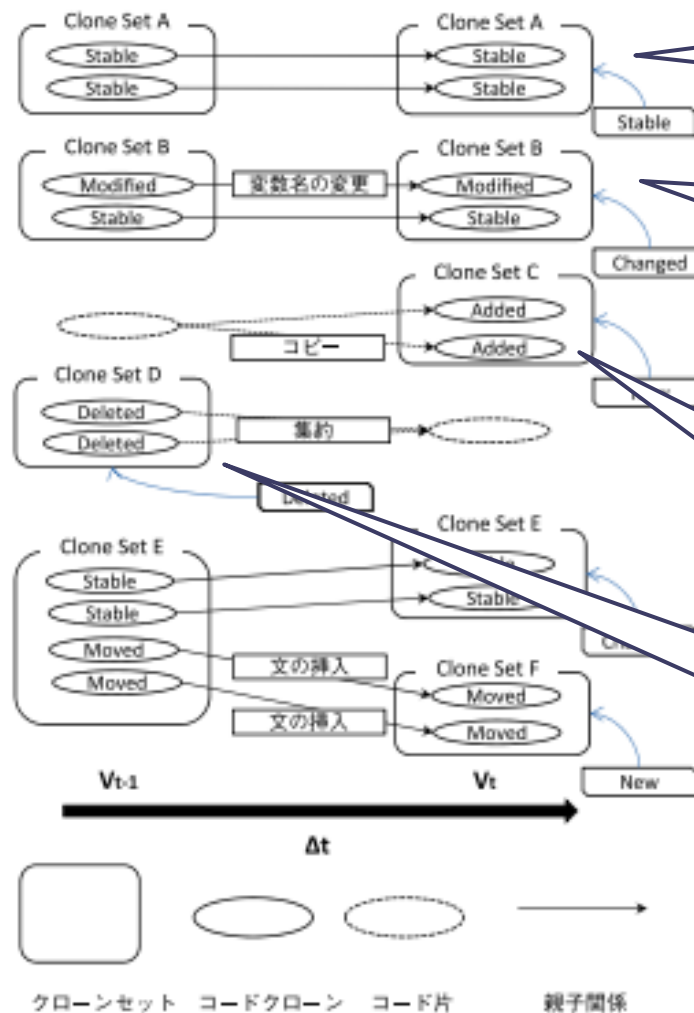


図 2 コードクローン・クローンセットの分類例

Fig. 2 Example of categorization of code clones and clone sets.

Stable Clone Set
クローンセットに属するコードクローンすべて。

Changed Clone Set
クローンセットのうち、修正されたコードクローン (Modified Clone) を含んでいる。

New Clone Set
 V_t のみに存在するクローンセット。
 V_{t-1} のあるコードからのコピーなどによって新たに発生したクローンセット。

Deleted Clone Set
 V_{t-1} のみに存在するクローンセット。
集約などによって除去されたクローンセット。

3、コードクローン変更管理システム

- システムの処理手順

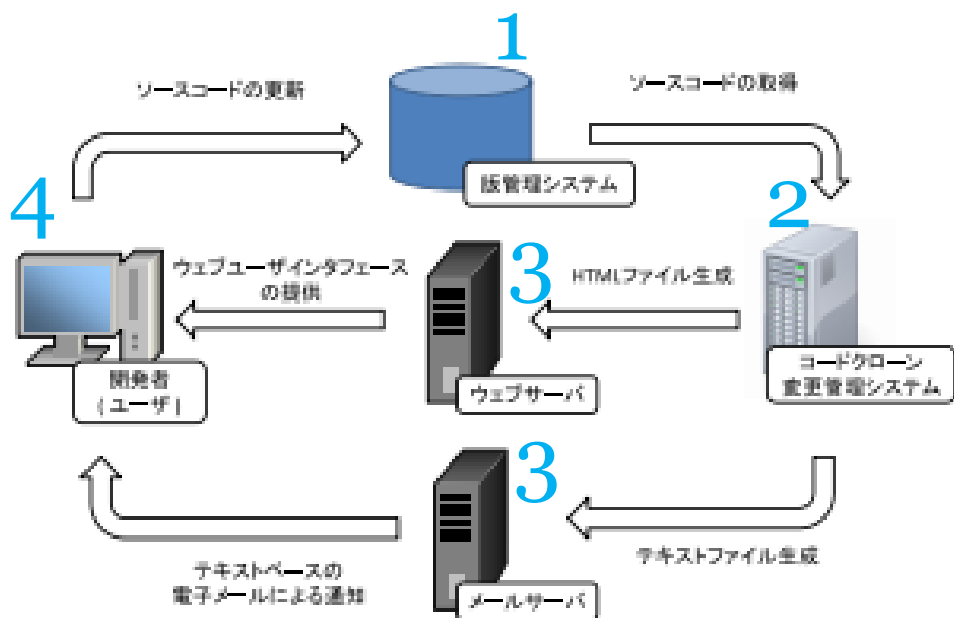


図 3 コードクローン変更管理システムの処理の流れ

Fig. 3 Process of code clone change management system.

1、版管理システムから最新バージョンのソースコードVtを取得。
(Vt-1は前回に取得済み)

2、クローンセットの分類。

3、分類結果に基づいて、HTMLファイルと電子メールによる通知のためのファイルの生成を行う。
(・電子メールを用いた通知)
(・ウェブインターフェースの提供)

4、開発者が変更されたコードクローンの存在を認識し、保守作業の必要性の判断を行う。

・電子メールを用いた通知

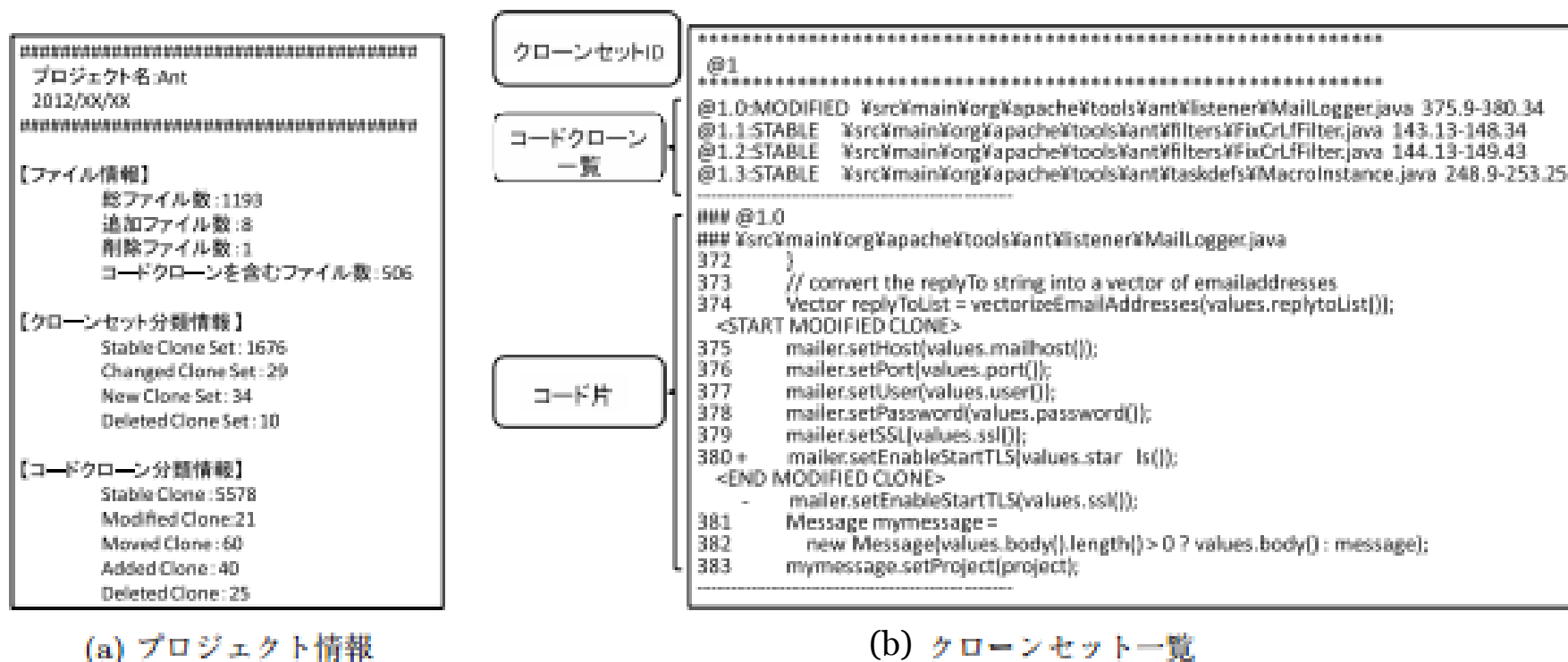


図 4 テキストファイルの出力例

Fig. 4 Example of text-based visualization.

(b)の説明

・ Changed Clone Set ・ New Clone Set ・ Deleted Clone Setに分類されたクローンセット一覧が出力される。これはChanged Clone Setの例である。

各々のクローンセットに関して・クローンセットID・クローンセットに属するコードクローン一覧・コードクローンとなっているコード片が出力される

ウェブインタフェースの提供

プロジェクト名: **Ant**

クローンセット一覧

Changed Clone Set				
クローンセット ID 66				
ID	分類	ファイル	位置	
79	DELETED	src\main\org\apache\tools\ant\filters\ExpandProperties.java	73.9-87.17	
117	STABLE	src\main\org\apache\tools\ant\filters\PrefixLines.java	86.9-100.17	
138	STABLE	src\main\org\apache\tools\ant\filters\SuffixLines.java	87.9-101.17	
クローンセット ID 44				
ID	分類	ファイル	位置	
232	MODIFIED	src\main\org\apache\tools\ant\filters\CrLfFilter.java	375.9-380.34	
82	STABLE	src\main\org\apache\tools\ant\filters\CrLfFilter.java	143.13-148.34	
82	STABLE	src\main\org\apache\tools\ant\filters\CrLfFilter.java	144.13-149.43	
823	STABLE	src\main\org\apache\tools\ant\filters\CrLfFilter.java	248.9-253.25	

(a) クローンセット一覧ページ

```

92  */
93  [START ID:79(Deleted Clone) CLONSET:39(Deleted Clone Set)]
94  public int read() throws IOException {
95  +   if (index > EOF) {
96  +     if (buffer == null) {
97  +       String data = readFully();
98  +
99  +       [START ID:79(Deleted Clone) CLONSET:66(Deleted Clone Set)]
100  +       int ch = -1;
101  +       if (queuedData != null && queuedData.length() == 0) {
102  +         queuedData = null;
103  +       }
104  +       if (queuedData != null) {
105  +         ch = queuedData.charAt(0);
106  +         queuedData = queuedData.substring(1);
107  +         if (queuedData.length() == 0) {
108  +           queuedData = null;
109  +         }
110  +       } else {
111  +         [END ID:78]
112  +         queuedData = readFully();
113  +         if (queuedData == null || queuedData.length() == 0) {
114  +           [END ID:79]
115  +           ch = -1;
116  +         } else {
117  +           Project project = getProject();
118  +           GetProperty getProperty =
119  +             if (propertySet == null) {
120  +               getProperty = PropertyHelper.getPropertyHelper(project);
121  +             } else {

```

(b) ソースファイルページ

図 5 ウェブユーザインタフェースの例

Fig. 5 Example of web-based user interface.

クローンセット一覧ページ

- 表示順序

Changed Clone Set, New Clone Set, Deleted Clone Set, Stable Clone Setの順。

- 各々の表示内容

「ID」「分類」「コードクローンが含まれるソースファイル名」「ソースファイル中のコードクロンの位置」。

※IDをクリックするとソースファイルページに遷移。

ソースファイルページ

コードクローンが含まれるソースファイルが表示される。

- 黄色の背景色

コードクローンとなっているコード。

- 「+」と「-」

「+」は追加行、「-」は除去行(灰色の背景色)

4、適用実験

- 目的

- ・ 開発者がこのシステムを用いて、保守作業が必要なコードクローンを発見することができるかを確認し、**有用性を評価**すること。

- ・ 評価実験において、保守作業が必要な**コードクローンの特徴を調査**すること。

• 実験内容

- 実験対象...日本電気株式会社のウェブアプリケーションソフトウェア開発(java)。ファイル数、約350ファイル。行数、約12万行。
- 期間...40日間
- CCFinderのトークンの閾値...30
- アンケート...対象者はjavaの経験が10年以上のプロジェクトマネージャ。アンケート内容は「保守作業の対象となったコードクロンの存在をすでにしっていたか」、「どのような保守作業を行ったか」。

• 結果と考察

表 1 保守作業の対象となったクローンセット

Table 1 Clone sets that required additional maintenance.

	検出日	保守実施日	分類	Q1	Q2	RAD(S)	LEN(S)	RNR(S)	NIF(S)	POP(S)	DFL(S)
1	'11/12/28	'12/01/05	New	NO	集約	0	141	78	1	2	141
2	'11/12/28	'12/01/05	New	NO	集約	1	36	83	2	2	36
3	'12/01/13	-	New	NO	集約	1	48	95	7	7	288
4	'12/01/13	-	New	NO	位置情報の記録	3	54	90	2	2	54
5	'12/01/13	-	New	NO	集約	0	52	61	1	2	52
6	'12/01/13	'12/01/17	New	NO	集約	0	57	94	1	2	57
7	'12/01/13	-	New	NO	集約	0	32	93	1	2	32
8	'12/01/16	-	New	NO	集約	0	37	83	1	2	37
9	'12/01/16	-	New	NO	集約	2	32	84	2	3	64
10	'12/01/18	-	New	NO	集約	0	53	90	1	2	53
11	'12/01/24	-	New	NO	集約	0	72	86	1	3	144

Q1の答えが全部NOなので、これらのクローンセットは新たに発見できたものということが分かる。

分類が全てNew(New Clone Set)のみになっている。このことから以前に保守作業が行われたコードクローンは含まれないということが分かる。

アンケート対象で集約の対象となったクローンセットの例

保守作業対象クローンセット1	
コードクローン1 : コンテンツの移動機能	コードクローン2 : コンテンツのコピー機能
<pre>for (inti = 0; i < contents.size(); i++) { try { Content content1 = contents.get(i); Content content2 = (Content) content1.clone(); content2.setTitle(StringU l.concatPath(toPath, content1.getName()), true); if (content1 instanceof Page) { movePage((Page) content1,(Page) content2); } else if (content1 instanceof File) { moveFile((File) content1,(File) content2); } else if (content1 instanceof Category) { moveCategory((Category) content1,(Category) content2); } } catch (Excep on e) { if (!(e instanceof AccessDeniedExcep on)) throw e; } }</pre>	<pre>for (inti = 0; i < contents.size(); i++) { try { Content content1 = contents.get(i); Content content2 = (Content) content1.clone(); content2.setTitle(StringU l.concatPath(toPath, content1.getName()), true); if (content1 instanceof Page) { copyPage((Page) content1,(Page) content2); } else if (content1 instanceof File) { copyFile((File) content1,(File) content2); } else if (content1 instanceof Category) { copyCategory((Category) content1,(Category) content2); } } catch (Excep on e) { if (!(e instanceof AccessDeniedExcep on)) throw e; } }</pre>
保守作業対象クローンセット2	
コードクローン1 : 最近参照されたコンテンツのリスト追加	コードクローン2 : ユーザに指定されたコンテンツのリスト追加
<pre>if ((recentContent != null) && (recentContent.size() > 0)) { Element element = new Element("div"); JDOMUtil.setAttribute(element, "class", ID.PLUGIN_RECENT); recentElement.addContent(recentContent); contents.add(recentElement); }</pre>	<pre>if ((htmlContent != null) && (htmlContent.size() > 0)) { Element element = new Element("div"); JDOMUtil.setAttribute(element, "class", ID.PLUGIN_RECENT); recentElement.addContent(htmlContent); contents.add(recentElement); }</pre>

呼び出すメソッド名だけが異なっている。

変数名が異なっている。

図 6 集約の対象となったクローンセットの例

Fig. 6 Example of a clone set that needs merging.

このようなメソッド名や変数名のみが異なっているクローンセットは、今後同時修正が必要となる可能性がある。

• コードクローンの特徴の調査

クローンセットSが与えられたときに以下のクローンセットメトリクスの抽出を行った。

- RAD(S)...コード片が含まれるファイル集合がファイルシステムの中で、ディレクトリ構造にどれだけ分散しているか。
- LEN(S)...コード片のトークン数の平均値。
- RNR(S)...コード片がどの程度非繰り返しであるかを表す。
- NIF(S)...コードクローンを所有するファイル数。
- POP(S)...コード片単位の要素数。
- DFL(S)...全コード片を集約した場合、減少が予測されるトークン数。

- マンホイットニーU検定を用いて調査

保守作業の対象のクローンセットと対象外のクローンセットのメトリクス値の分布に対して違いがあるかをマンホイットニーU検定で調査。

表 1 保守作業の対象となったクローンセット

Table 1 Clone sets that required additional maintenance.

	検出日	保守実施日	分類	Q1	Q2	RAD(S)	LEN(S)	RNR(S)	NIF(S)	POP(S)	DFL(S)
1	'11/12/28	'12/01/05	New	NO	集約	0	141	78	1	2	141
2	'11/12/28	'12/01/05	New	NO	集約	1	36	83	2	2	36
3	'12/01/13	-	New	NO	集約	1	48	95	7	7	288
4	'12/01/13	-	New	NO	位置情報の記録	3	54	90	2	2	54
5	'12/01/13	-	New	NO	集約	0	52	61	1	2	52
6	'12/01/13	'12/01/17	New	NO	集約	0	57	94	1	2	57
7	'12/01/13	-	New	NO	集約	0	32	93	1	2	32
8	'12/01/16	-	New	NO	集約	0	37	83	1	2	37
9	'12/01/16	-	New	NO	集約	2	32	84	2	3	64
10	'12/01/18	-	New	NO	集約	0	53	90	1	2	53
11	'12/01/24	-	New	NO	集約	0	72	86	1	3	144

表 2 マンホイットニー U 検定の結果

Table 2 Result of Mann-Whitney U test.

	RAD(S)	LEN(S)	RNR(S)	NIF(S)	POP(S)	DFL(S)
<i>p</i> 値	0.5106	0.2419	0.0368	0.5955	0.2793	0.2283
帰無仮説	採択	採択	棄却	採択	採択	採択

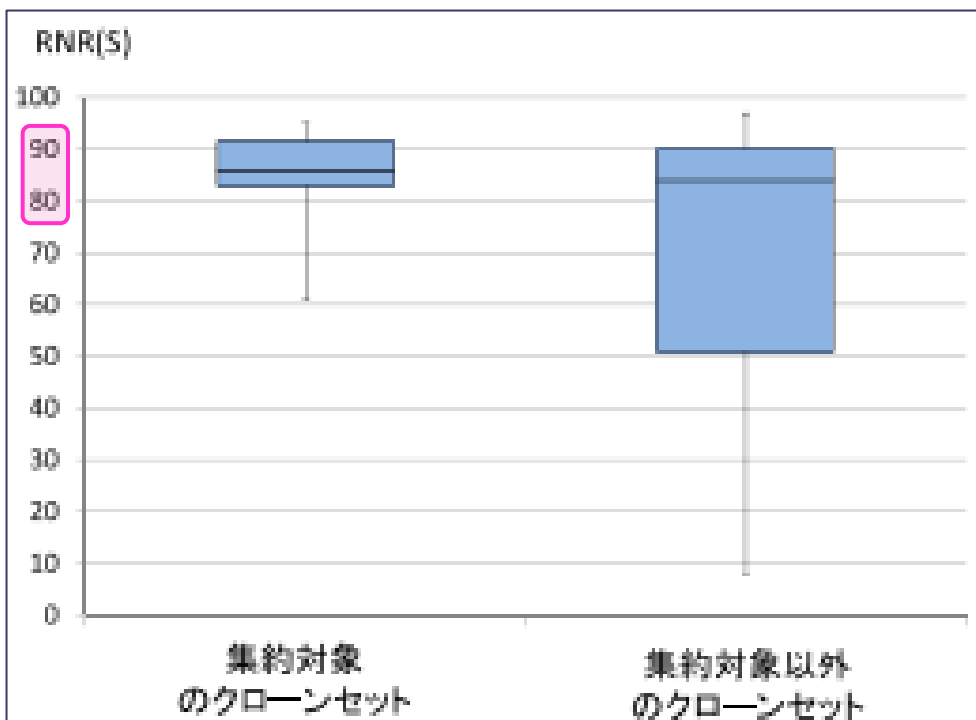


図 8 RNR (S) メトリックの比較

Fig. 8 Comparison of RNR (S) metric.

RNRの値は非繰り返し率なので低いほうが保守を行う必要がないもの。

この結果を見ると、RNRの値は高く保守を行う必要がある結果となっている。

マンホイットニーU検定の結果では棄却しているのでRNRをフィルタリングすることによって、さらに集約対象となる可能性の高いクローンセットを提示することが可能になると考えられる。