

コードクローンの計測とEPM

2004年10月5日

奈良先端科学技術大学院大学
文部科学省EASEプロジェクト
門田 暁人

コードクローンとは

ソースコード中の類似したコード片

.....

.....

.....

.....

.....

.....

→

```
r = top - z;
if (r > 0) n = 1;
h = getsum(r);
```

.....

.....

.....

.....

.....

.....

→

```
r = top - y;
if (r > 0) n = 0;
h = getsum(r);
```

プログラム

クローンが生じる理由:
コピー&ペースト, 定型処理, 意図的な繰り返し, 偶然, プログラミング言語に適切な機能がないため, など ²

コピー&ペーストによる増殖

資料-4

コピー

モジュールA

コピー

モジュールB

コピー

モジュールC

大規模ソフトウェアの多くの部分がコードクローンであると報告されている。(5~50%). ³

本発表の概要

- EPMによるプロジェクト管理を行う場合に、コードクローン計測が有効である。
 - より簡便な方法として、ソースコードの圧縮率を計測する方法もある。
- コードクローンの種類, 許容量

4

動機 (1)

SLOC (ソースコードの行数)は有効な指標ではあるがプロジェクト管理を行うには不十分である。

Version	SLOC (approx.)
442	1500
450	2500
500.rc1	3500
500.rc5	3800
500	4000
505	4500
516	5500
517.1	5800
520.2	6500
525	7000
528	7500
530	7000

大きな進捗があった?

逆行?

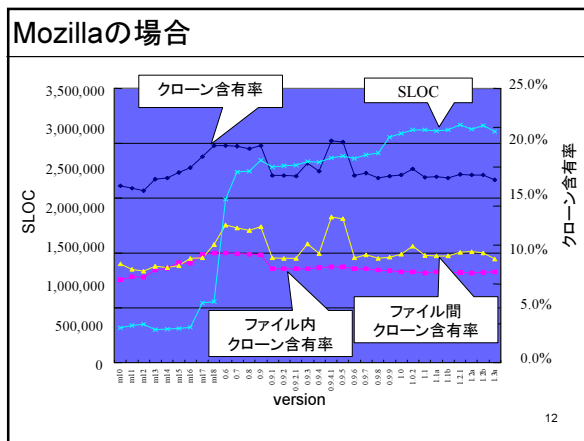
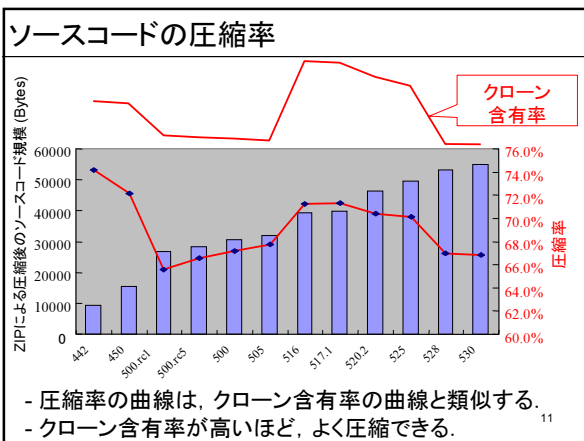
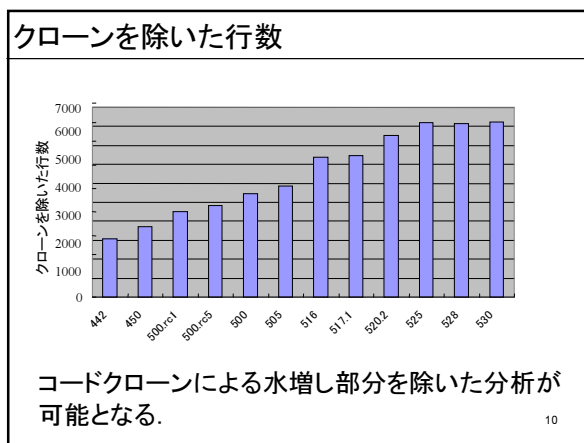
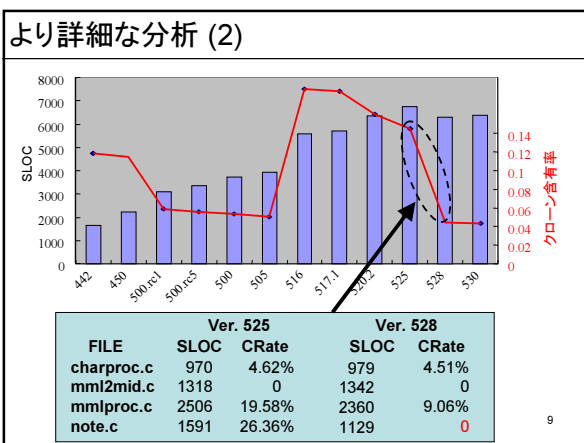
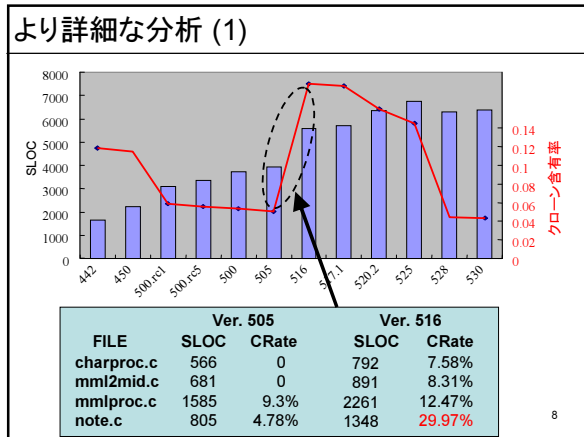
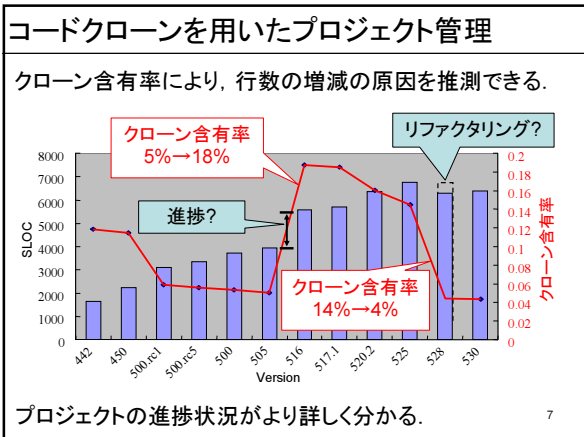
プロジェクトの進捗は、行数だけでは分からない。 ⁵

動機 (2)

しばしば、次のような話を聞きます ...

- 協力会社がコピー&ペーストによってソースコードの行数の水増しを行い、見かけの仕事量(行数/月)を増やしている。
- 開発者がデバッグコードを挿入したり削除することがことで、行数が急に増えたり減ったりする。

プロジェクト管理者は、行数に代わる指標を必要としている。 ⁶



コードクローンと保守性

・コードクローンはソフトウェアの保守性を低下させる一因であると考えられている。

修正を加える
要修正
要修正
要修正

モジュールA モジュールB モジュールC

クローン部分に修正を加える場合には、関連する全てのクローンを修正する必要がありコストがかかる。 13

コードクローンと信頼性

・コードクローンはソフトウェアの信頼性を低下させる可能性がある。

修正済
修正済
修正済
修正し忘れ

モジュールA モジュールB モジュールC

修正し忘れたクローンにバグが残っていると、信頼性は低下する。 14

コードクローンに関する疑問

大規模な開発では、コードクローンを全く含まないソフトウェアを作ることは困難であると思われる。

しかし、

- ・どのような種類のクローンであれば許容され、どのようなものは許容されないのか不明である。
- ・ある程度のコードクローンが許容されるとしても、どのくらいの量のコードクローンなら許容範囲であるか不明である。

↓

多数のオープンソースプログラムを分析し、クローンの実態を調査した。 15

クローン含有率

・GNUプロジェクトWebサイトにある125個のC言語プログラムを対象としてコードクローン含有率を計測した。

・50トークン(約21行)以上の一致列をクローンとして検出した。

	ファイル数	行数	クローン含有率	ファイル内クローン含有率	ファイル間クローン含有率
平均値	86	55,675	11.3%	9.1%	3.0%
最大値	3755	2,678,939	61.2%	61.2%	37.6%
最小値	1	478	0	0	0

ソースコードの公開を前提としたオープンソースソフトウェアであってもクローンを多く含むものが存在する。 16

クローン含有率の高いソフトウェア

	ファイル数	行数	クローン含有率	ファイル内クローン含有率	ファイル間クローン含有率	主な要因
superopt	1	3,080	61.2%	61.2%	0	(1)
chemtool	12	27,254	51.6%	26.5%	37.6%	(2)
gcc	19	9,332	47.0%	47.0%	0	(2)
bucob	43	44,828	43.7%	9.9%	35.5%	(2)
xcdroast	17	36,193	42.2%	39.4%	15.5%	(3)
rubrica	57	30,542	35.3%	34.1%	10.7%	(3)
gaby	106	60,652	33.3%	27.4%	15.9%	(2)
calculator	8	2,999	32.5%	32.5%	0	(2)
pitchtune	10	4,373	31.9%	31.9%	0	(2)
hme	17	7,211	30.9%	26.6%	7.4%	(4)

主なクローン発生要因: (1)プラットフォーム依存コード, (2)自動生成コード, (3)GUIに関するコード, (4)コピー&ペーストによるコード₁₇

要因1: 多プラットフォーム対応のためのコード

```

superopt:
    case LSHIFTR_CO:
        printf("shrl %s,%s",NAME(s2),NAME(d));break;
    case ASHIFTR_CO:
        printf("sarl %s,%s",NAME(s2),NAME(d));break;
    case SHIFTL_CO:
        printf("shll %s,%s",NAME(s2),NAME(d));break;
    case ROTATEL_CO:
        printf("roll %s,%s",NAME(s2),NAME(d));break;
    #elif ALPHA
        ...
    #elif HPPA
        ...
    #elif SH
        ...
    
```

CPUを指定して数式を入力すると、そのCPUに対応したアセンブリコードを出力する。

・CPU毎の処理がコードクローンとなっている。

・各CPUに依存した処理と非依存の処理の切り分けはうまく行われており、コードクローンが保守性を低下させているとはいえない。 18

要因2: 自動生成コード

- ・ プログラムトランスレータによるクローン
Pascal-to-C translatorなど
- ・ GUIビルダツールによるクローン
GLADEなど
- ・ 字句・構文解析器ジェネレータによるクローン
Flex, Bisonなど

自動生成されたコードは、人間が編集することは想定されておらず、コードクローンが保守性を低下させているとは言えない。 19

要因3: GUIに関するコード

```
tbl = gtk_table_new(3,8,TRUE);
gtk_table_set_row_spacings(GTK_TABLE(tbl),10);
gtk_table_set_col_spacings(GTK_TABLE(tbl),10);
gtk_box_pack_start(GTK_BOX(vbox),tbl,FALSE,FALSE,10);
gtk_widget_show(tbl);
l1 = rightjust_gtk_label_new(text(103));
gtk_table_attach_defaults(GTK_TABLE(tbl),l1,0,2,0,1);
gtk_widget_show(l1);
```

Clone pair

```
tbl = gtk_table_new(2,16,TRUE);
gtk_table_set_row_spacings(GTK_TABLE(tbl),10);
gtk_table_set_col_spacings(GTK_TABLE(tbl),10);
gtk_box_pack_start(GTK_BOX(actionspace),tbl,FALSE,FALSE,10);
gtk_widget_show(tbl);
l1 = rightjust_gtk_label_new(text(219));
gtk_table_attach_defaults(GTK_TABLE(tbl),l1,0,6,0,1);
gtk_widget_show(l1);
```

- ・ GUI部品を生成するための定型コード群がクローンとなっている。
- ・ 保守性への悪影響は小さい。
- ・ 継承やラッパー関数によりクローンを減らすことは可能。 20

要因4: コピー&ペーストによるコード

容易に共通化できるものと、そうでないものがある。

- ・ 容易に共通化できないもの
多数のローカル変数を含むコード
ローカル変数への多数の代入文を含むコード
- より大域的な設計の見直しが望ましい。

21

クローン含有率の基準値

クローン含有率		自動生成コード	
		有	無
GUI	有	16.9%	13.8%
	無	13.4%	7.2%

ファイル内クローン含有率		自動生成コード	
		有	無
GUI	有	13.6%	11.7%
	無	7.9%	6.1%

ファイル間クローン含有率		自動生成コード	
		有	無
GUI	有	4.9%	3.4%
	無	6.6%	1.3%

22

関連研究

分析者	システム	言語	行数	基準	クローン含有率
Baker	AT&T System	C	200,000	30LOC	23%
Baxter	プロセス制御	C	400,000	n.s.	12.7%
Ducasseら	DBサーバ	Smalltalk	245,000	10LOC	36.4%
Ducasseら	給与計算	COBOL	40,000	10LOC	59.3%
Mondenら	MIS	拡張COBOL	1,000,000	30LOC	42.6%
本研究	125個のオープンソース	C	—	50Token (21LOC)	11.3%

- ・ 商用ソフトの方が、オープンソースよりもクローンが多い。
- ・ 特に、COBOL系はクローンが多い。
- ・ システムの種類の違い、言語の違いは考慮されるべき。

保守作業へのインタビュー結果

- ・ Q: クローンを意図的に作ることはありますか。
- A: あります。機能追加を行う際に、安全に変更できそうにない場合、コピー&ペーストして作成したクローンに手を加えます。(元コードの信頼性は保たれます。)
- ・ Q: クローンをどう管理してますか。
- A: 「横並びチェック表」を使います。この表には、ソフトウェア中の関連のある部分が掲載されています。ソフトウェアに変更を加えた場合には、この表を見て、関連のある部分が他にないかどうかをチェックします。(クローンもここでチェックされます。)

24

まとめ

- ソースコードの水増し(コピー&ペースト)によりコードクローンが発生する.
- プロジェクト管理を行う場合に, コードクローン計測が有効である.
 - ソースコードの圧縮率を計測する方法もある.
- 一般に, コードクローンは保守性を低下させるが, クローンの全く無いソフトを作るのは不可能であるし, 全てのクローンが悪いわけではない. 基準値を大幅に上回らないことが重要.

25

CloneWarriorのデモ

