



CBCD: Cloned Buggy Code Detector

Jingyue Li Michael D. Ernst

ICSE '12 Proceedings of the 34th International Conference on Software Engineering
Pages 310-320



目次

- 背景・目的
- 複製されたバグの調査
- CBCD の構築
- CBCD の評価
- まとめ・所感



背景·目的

背景

- 開発者はコピー&ペーストをする中でバグを複製することがある
- 複製されたバグの一方を見つけて修正しても、もう一方のバグが後に見つかり修正する必要が生じる



目的

- どれだけバグが複製されているかを調査する
- 複製されたバグを検出するツール CBCD を構築する



複製されたバグの調査

調査概要

- データセット
 - Linux Kernel , Git , PostgreSQL , 商用バグレポートシステム
- それぞれのプロジェクトで複製されたバグを目視で探す

Linux Kernel

コミットメッセージやBTSの中でそのキーワードを含むものを探した

TABLE I. CLONED BUGS WHICH EXIST IN MORE THAN ONE PLACE IN THE LINUX KERNEL

Key words used for searching the SCM	Number of distinct bugs existing in more than one place	Number of bugs whose clones we can locate
same bug	53	23
same fix	48	24
same issue	62	39
same error	7	6
same problem	112	65
Sum	282	157

TABLE II. EXAMPLES OF CLONED BUGS IN THE LINUX KERNEL

Phrases in the SCM explaining the cloned bugs	Code modified (i.e., the lines of code modified by the bug fix)
This is quite the same fix as in 2cb96f86628d6e97fcbda5fe4d8d74876239834c	<pre>static int my_atoi(const char *name){ int val = 0; for (;;) name++ { switch (*name) { case '0' ... '9': val = 10*val+(*name-'0'); break; default: return val; } }}</pre>
This patch fixes iwl3945 deadlock during suspend by moving notify_mac out of iwl3945 mutex. This is a portion of the same fix for iwlwifi by Tomas.	<pre>ieee80211_notify_mac(priv->hw, IEEE80211_NOTIFY_RE_ASSOC);</pre>
It turns out that at least one of the caller had the same bug.	<pre>ret = btrfs_drop_extents(trans, root, inode, start, aligned_end, start, &hint byte);</pre>
Other platforms have this same bug, in one form or another	<pre>atomic_inc(&call_data->finished); func(info);</pre>

Git と PostgreSQL

TABLE III. CLONED BUGS WHICH EXIST IN MORE THAN ONE PLACE IN GIT AND POSTGRESQL

Key words used for searching the SCM	GIT		POSTGRESQL	
	<i>Number of distinct bugs existing in more than one place</i>	<i>Number of bugs whose clones we can locate</i>	<i>Number of distinct bugs existing in more than one place</i>	<i>Number of bugs whose clones we can locate</i>
same bug	7	5	9	9
same fix	7	4	5	4
same issue	14	3	2	0
same error	0	0	1	8
same problem	5	0	16	1
Sum	33	12	33	22

商用バグレポートシステム

- 3.8 % (969/25420) のバグが複製されていることが分かった

TABLE IV. CLONED BUGS WHICH EXIST IN MORE THAN ONE PLACE IN THE COMMERCIAL SOFTWARE PRODUCT LINE

Key words used for searching the bug reports	Number of distinct bugs existing in more than one place
same bug	170
same fix	40
same issue	302
same error	56
same problem	401
Sum	969



CBCD の構築

CBCD の概要

- プログラム依存グラフに変換されたコードから重複するバグを検出するツール
- 重複するバグを検出する際, バグと同じ形の部分グラフをシステム全体から探索する
- グラフ同型問題であり, 検出に指数時間を要するため検出前に最適化処理を行う

プログラム依存グラフ: 命令をノードとし, データ依存関係や制御依存関係をエッジで表した有向グラフ

CBCD の概要

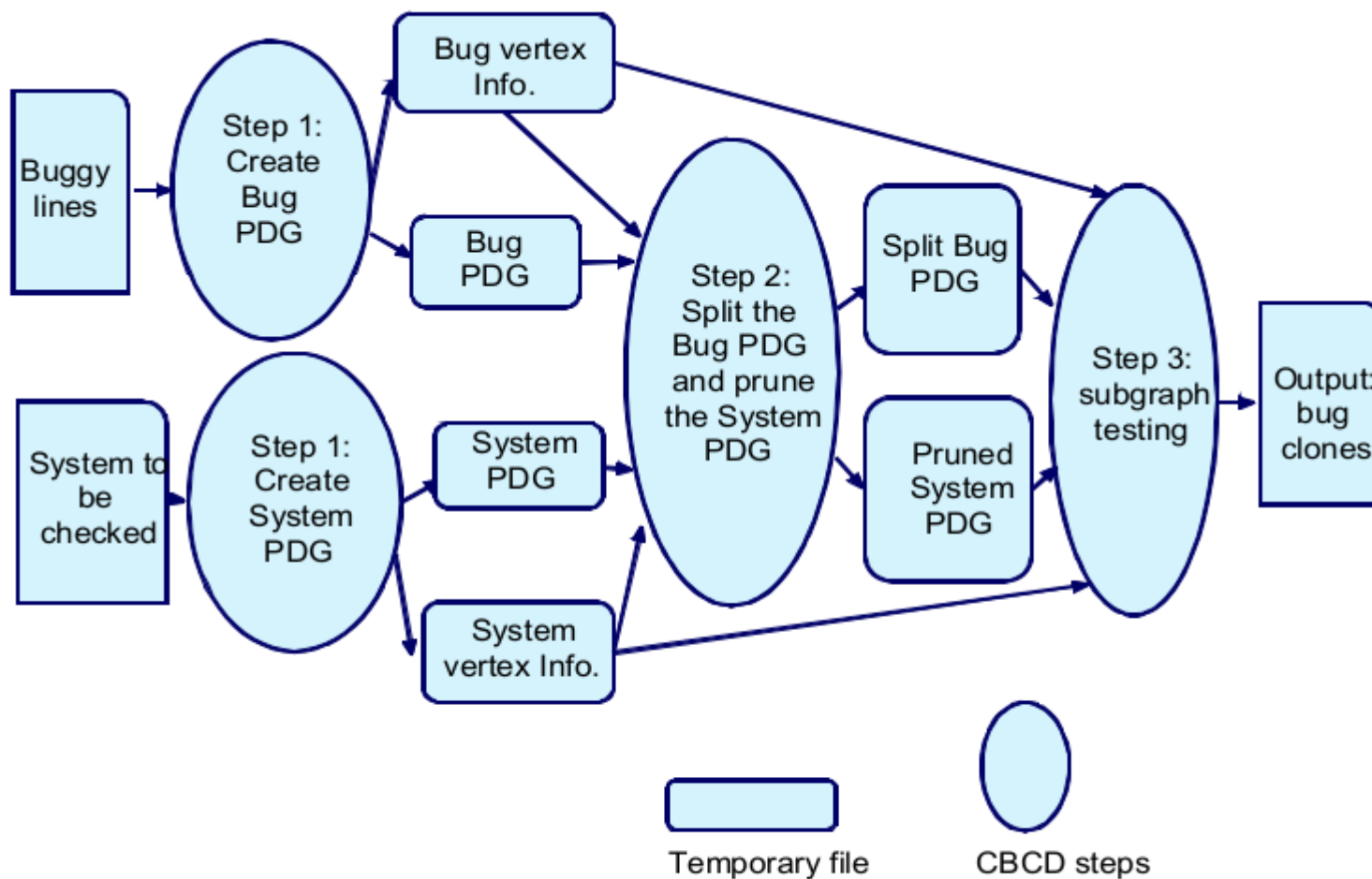


Figure 1. Architecture of CBCD

最適化処理

1. システム PDG から無関係なノードやエッジを取り除く
2. システム PDG を切り分ける
3. 無関係な PDG を取り除く
4. 大きいバグ PDG を切り分ける



CBCD の評価

評価実験の概要

- 実験対象のプロジェクトは Linux Kernel (数百万行), Git (約300万行), PostgreSQL (約100万行)
- 他のクローン検出ツールと検出精度や検出速度を比較する
 - Simian : 行単位の検出ツール
 - CCFinder : 字句単位の検出ツール
 - Deckard : 抽象構文木単位の検出ツール
 - CloneDR : 抽象構文木単位の検出ツール

検出精度の比較

- メトリクス

- N1 : no false positives, no false negatives
- N2 : no false positives, some false negatives
- N3 : some false positives, no false negatives
- N4 : some false positives, some false negatives

TABLE VI. COMPARISON WITH OTHER CODE CLONE DETECTORS

	CBCD	Simian	CCFinder	Deckard	CloneDR -slim
N1	36 (68%)	16 (30%)	24 (45%)	14 (26%)	31 (58%)
N2	6 (11%)	36 (68%)	11 (21%)	29 (55%)	14 (26%)
N3	11 (21%)	1 (2%)	12 (23%)	6 (11%)	7 (13%)
N4	0 (0%)	0 (0%)	6 (11%)	4 (8%)	1 (2%)

検出速度の比較

TABLE XII. RUNNING TIME OF EACH STEP OF CBCD

Id	NLOC / Number of PDG edge		CBCD steps		
	Sys.	Bug	1	2	3
Git-1	67K/358K	10/38	6m	13s	5s
Git-2	75K/441K	4/4	15m	4s	2s
Git-3	81K/414K	9/39	18m	9s	3s
Git-4	81K/414K	16/33	18m	6s	2s
Linux1	170K/1022K	6/70	32m	15s	6s
Linux2	140K/830K	3/3	25m	16s	4s
Linux3	363K/1970K	4/4	159m	39s	8s
Linux4	313K/1645K	3/13	95m	17s	7s

TABLE XIII. RUNNING TIME OF OTHER CLONE DETECTORS

Id	Simian	CCFinder	Deckard
Git-1	2s	5m	4m
Git-2	2s	6m	5m
Git-3	2s	8m	6m
Git-4	2s	8m	6m
Linux1	6s	63m	8m
Linux2	5s	34m	7m
Linux3	16s	899m	32m
Linux4	13s	623m	24m



まとめ・所感

まとめ

- 複製されたバグは商用ソフトウェアで 3.8% と少なく, OSS でも規模に対してかなり少ない
- 最適化処理により実用的な時間で複製されたバグの検出が出来た

所感

- 最適化処理さえうまくやれば指数時間要する問題も実用的な時間で解くことが出来るのは参考になった
- 検出精度が他のツールと少し競っているのが気になった