

Relationship between the Effectiveness of Spectrum-Based Fault Localization and Bug-Fix Types in JavaScript Programs

Béla Vancsics, Attila Szatmári, Árpád Beszédes

JavaScriptプログラムにおけるSBFLの有効性と
バグ修正タイプの関係について

SANER 2020

発表者：辻 裕真

背景

- ソフトウェアの障害の特定は、メンテナンスやソフトウェアの進化において重要なプロセスである.
- SBFL(Spectrum-Based Fault Localization)はソフトウェアの障害位置を特定するための手法としてよく知られており、その有効性に関する多くの研究が行われてきたが、そのほとんどがJavaおよびC/C++プログラムに関するもので、JavaScriptに関する研究は行われていない.
- この研究ではJavaScriptにおけるSBFLの有効性を調査すること、さまざまなSBFLアルゴリズムのパフォーマンスを調べることを目的としている.

Research Questions

RQ1 : Tarantula, Ochiai, DStarの3つのSBFLアルゴリズムでJavaScriptの障害特定に関してどのように異なるか.

RQ2 : 3つのアルゴリズムでより良い, または悪い動作をするバグ修正タイプはあるか.

研究手法

データセット

- ・ バグ関連のソフトウェアエンジニアリング研究のために作成されたJavaScriptバグデータセットであるBugsJSに含まれる453のバグから, 336のバグを使用. (テストの実行を高速に行うために行なった変更により使用できなくなったものを除いた.)
- ・ 全てのバグについて修正内容に応じてラベリングを行い, カバレッジ(テスト網羅率)測定と障害特定スコアの計算などを行う.

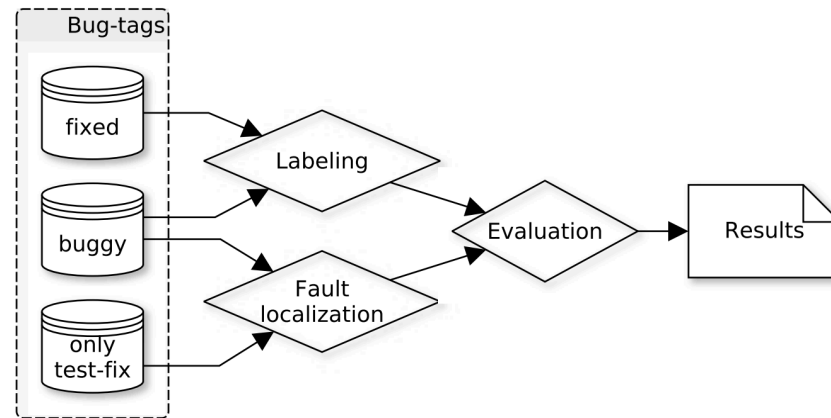


Fig. 1. Experiment overview

研究手法

ラベリング

TABLE II
CATEGORIES PROPOSED BY PAN ET AL.

Bug fix patterns	Description
IF	If-related: Changes to if conditions in any way.
MC	Method call: Changes that include calling a method.
SQ	Sequence: Three or four changes in an operation sequence.
LP	Loop: Change either in the loop predicate or in the scope of the loop statement.
AS	Assignment: The expression on the right hand side changes but the expression on the left stays the same.
SW	Switch: Additional or removal of switch branch
TY	Try-related: Additional or removal of a try or a catch block
MD	Method declaration: Addition or removal of a method declaration, or a change of method declaration
CF	Class field: Addition, removal or change of Class fields.

最も一般的な修正パターンはIFとMCであった。(Panらの研究による)

研究手法

障害特定のランク計算

$$COV = \begin{bmatrix} c_{1,1} & \dots & c_{1,a} \\ \vdots & \ddots & \vdots \\ c_{b,1} & \dots & c_{b,a} \end{bmatrix}, \quad RES = \begin{bmatrix} r_1 \\ \vdots \\ r_b \end{bmatrix},$$

$$c_{\tau,\phi} \in \{0,1\} : \tau \in T \ \phi \in F, \\ c_{\tau,\phi} = 1 \text{ if } \phi \text{ is covered by } \tau,$$

$$r_{\tau} \in \{0,1\} : \tau \in T, \\ r_{\tau} = 0 \text{ if } \tau \text{ is pass, else } 1.$$

COVは行がテスト，列がコード要素(関数など)を表し，コード要素 ϕ がテストケース τ でカバーされている場合値は1となり，それ以外は0となる.

RESはテストの結果を表すベクトルでテストに合格した場合は0となり，それ以外は1となる.

研究手法

障害特定のランク計算

ϕ_{ep} : number of passing tests covered by ϕ

ϕ_{ef} : number of failing tests covered by ϕ

ϕ_{np} : number of passing tests not covered by ϕ

ϕ_{nf} : number of failing tests not covered by ϕ

DStar [25] :
$$\frac{\phi_{ef}^2}{\phi_{ep} + (\phi_{ef} + \phi_{nf}) - \phi_{ef}},$$

Ochiai [24]:
$$\frac{\phi_{ef}}{\sqrt{(\phi_{ef} + \phi_{nf}) \cdot (\phi_{ef} + \phi_{ep})}},$$

Tarantula [23]:
$$\frac{\frac{\phi_{ef}}{\phi_{ef} + \phi_{nf}}}{\frac{\phi_{ef}}{\phi_{ef} + \phi_{nf}} + \frac{\phi_{ep}}{\phi_{ep} + \phi_{np}}}.$$

COVとRESから関数 ϕ について4つの基本的なメトリクスを計算し, 3つのアルゴリズムごとにランクを計算する.

結果

RQ1 : Tarantula, Ochiai, DStarの3つのSBFLアルゴリズムでJavaScriptの障害特定に関してどのように異なるか.

TABLE III
AVERAGE RANKS

Project	Tarantula	Ochiai	DStar
Bower	25.83	19.17	17.50
Shields	5.83	5.83	5.17
Hexo	3.25	3.00	80.88
Hessian.js	4.81	3.88	3.00
Express	8.10	7.94	7.94
Pencilblue	1.83	1.67	1.67
Eslint	20.39	19.90	19.90
All	18.24	17.73	20.47

- 表から, 異常値を除いて3つのアプローチのほとんどが同様の結果であったことが分かる.
- HexoプロジェクトでのDStarに関して, このアルゴリズムはいくつかのケースで極端なランクを生成してしまうことが他の研究で既に分かっている.

結果

RQ1 : Tarantula, Ochiai, DStarの3つのSBFLアルゴリズムでJavaScriptの障害特定に関してどのように異なるか.

TABLE III
AVERAGE RANKS

Project	Tarantula	Ochiai	DStar
Bower	25.83	19.17	17.50
Shields	5.83	5.83	5.17
Hexo	3.25	3.00	80.88
Hessian.js	4.81	3.88	3.00
Express	8.10	7.94	7.94
Pencilblue	1.83	1.67	1.67
Eslint	20.39	19.90	19.90
All	18.24	17.73	20.47

- その外れ値を除くと, DStarは3つのアルゴリズムで最も良い結果であるといえる.
- 結果として, OchiaiとDStarには大きな違いはなかったがTarantulaは少し悪かった.

結果

RQ1 : Tarantula, Ochiai, DStarの3つのSBFLアルゴリズムでJavaScriptの障害特定に関してどのように異なるか.

TABLE IV
TOP-N RANKS

	Tar.	(%)	Ochiai	(%)	DStar	(%)
top-1	96	28.6	101	30.1	100	29.8
top-3	206	61.3	214	63.7	214	63.7
top-5	255	75.9	264	78.6	264	78.6
top-10	300	89.3	306	91.1	303	90.2
other	36	10.7	30	8.9	33	9.8

- 表はアルゴリズムごとにいくつかのバグが各ランク(修正の優先度)に振り分けられたかを表している.
- 開発者はtop-5または多くてもtop-10までの要素のみを調査する傾向がある.

結果

RQ2：3つのアルゴリズムでより良い，または悪い動作をするバグ修正タイプはあるか.

TABLE V
OVERALL BUG-FIX TYPE STATISTICS BASED ON TARANTULA

Bug-fix type	Bower	Shields	Hexo	Hessian	Express	Pencilblue	Eslint	Total
IF	2	0	5	4	12	2	151	176
AS	2	3	9	2	9	2	75*	102*
MD	0	0	0	0	1	0	23	24
LP	0	0	0	0	2	0	6	8
MC	0	0	1	0	3	1	60	65
SQ	0	0	1	3	6	0	19	29
SW	0	0	0	0	1	0	4	5
TY	0	0	0	0	0	0	1	1
CF	0	1	0	0	0	0	1	2
Total	4	4	16	9	34	5	340*	

- 表はプロジェクトごとのバグ修正タイプの分類である.
- ラベルの数(412)がバグの数(336)より多いのは同じバグに複数のラベルが付けられることがあるためである.

結果

RQ2：3つのアルゴリズムでより良い，または悪い動作をするバグ修正タイプはあるか.

TABLE VI
NUMBER OF LABELS (PER METRICS)

Name	TOP-1 (#)			TOP-3 (#)			TOP-5 (#)			TOP-10 (#)			OTHER (#)		
	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar
IF	57	60	58	115	119	119	142	147	146	164	166	164	12	10	12
AS	29	31	30	58	63	62	73	76	74	87	92	89	15	11	14
CF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MD	7	7	7	17	18	19	19	20	20	21	22	22	3	2	2
MC	24	25	25	40	41	41	46	47	47	58	57	59	7	8	8
SQ	3	3	3	14	14	16	19	20	21	23	24	24	6	5	5
SW	1	1	1	3	3	3	3	4	4	4	4	4	1	1	1
LP	0	0	0	4	4	4	7	7	7	7	7	7	1	1	1
TY	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0
All	122	128	125	253	264	266	311	323	321	366	374	369	46	39	44

結果

RQ2：3つのアルゴリズムでより良い，または悪い動作をするバグ修正タイプはあるか.

TABLE VII
PERCENTS OF LABELS (PER METRICS)

Name	TOP-1 (%)			TOP-3 (%)			TOP-5 (%)			TOP-10 (%)			OTHER (%)		
	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar	Tarantula	Ochiai	DStar
IF	32.4	34.1	33.0	65.3	67.6	67.6	80.7	83.5	83.0	93.2	94.3	93.2	6.8	5.7	6.8
AS	28.4	30.1	29.1	56.9	61.2	60.2	71.6	73.8	71.8	85.3	89.3	86.4	14.7	10.7	13.6
CF	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0	50.0
MD	29.2	29.2	29.2	70.8	75.0	79.2	79.2	83.3	83.3	87.5	91.7	91.7	12.5	8.3	8.3
MC	36.9	38.5	38.5	61.5	63.1	63.1	70.8	72.3	72.3	89.2	87.7	87.7	10.8	12.3	12.3
SQ	10.3	10.3	10.3	48.3	48.3	55.2	65.5	69.0	72.4	79.3	82.8	82.8	20.7	17.2	17.2
SW	20.0	20.0	20.0	60.0	60.0	60.0	60.0	80.0	80.0	80.0	80.0	80.0	20.0	20.0	20.0
LP	0.0	0.0	0.0	50.0	50.0	50.0	87.5	87.5	87.5	87.5	87.5	87.5	12.5	12.5	12.5
TY	0.0	0.0	0.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	0.0	0.0	0.0

- ・ IFタイプのバグは高いランクがつけられやすいことが分かる.
(top-10まででほとんどのバグが発見されている)
- ・ SQタイプは高いランクで発見される可能性が低い.

議論

- Tarantulaが最初に公開されたSBFLメソッドの1つであるため、他の2つと比べパフォーマンスが低いことは仕方がない.
- DStarは全体的に最良の結果を示していたが、これらのアルゴリズムの中で明確に1番であると主張することはできない.
- SQタイプの位置特定が難しく、IFタイプの位置特定が簡単である理由は今のところ不明である.

結論

- ・ 3つのSBFLアルゴリズムはそれぞれほぼ同様の結果を示しており、最も良いものを決めることはできない.
- ・ バグの修正タイプごとに障害位置特定の有効性の違いを確認することができた.