

Predicting aging-related bugs using software complexity metrics

Jason BURY

Faculté des Sciences
Université de Mons



2018 年 9 月 2 日



Aging-Related Bugs

Mandelbug

If

- A time lag between the fault activation and the failure occurrence.
- Bug influenced by timing of operations or order of operations or environment interactions.

Then \implies *Mandelbug* :-(
 Else \implies *Bohrbugs* :-) , can thus be easily isolated



Aging-Related Bugs 试试中文

Aging-related mandelbug

If Mandelbug and

The bug causes the accumulation of internal error states

Or

The running time increase the risk of the bug activation/propagation

Then \implies *Aging-related Mandelbug* : '(

Also called *Aging-related bug* or **ARB**.

Aging-Related Bugs

Examples

What can causes an ARB ?

- Numerical error due to rounding or imprecision accrue over time
- A malloc() without free()
- A queue continuously increased
- Size of ressource exceeded
- ...

How can we detect them ?

short answer: **Select metrics then use a classifier !**

But, there are remaining questions about that solution:

Questions

- Which metrics to use ?
- New metrics ?
- Which classifier to use ?
- will be the classifier effective for all projects ?

Project used as training set

3 complex projects will be used.

- Linux
- MySQL
- CARDAMOM, a middleware for air traffic control systems

ARBs are found manually.



Which metrics to use ?

46 metrics correlated with bug density:

- Program size. 26 metrics
(lines of codes, comments, blank, semicolons, functions, preprocessor)
- Cyclomatic complexity. 11 metrics
- Halstead metrics. 9 metrics



Halstead metrics

Halstead metrics: set of metrics based on the number of operands and number of operators.

Based on

η_1 = The number of distinct operators

η_2 = The number of distinct operands

N_1 = The total number of operators

N_2 = The total number of operands

Some halstead metrics

- Program vocabulary = $\eta = \eta_1 + \eta_2$
- Program length = $N = N_1 + N_2$
- Volume = $N \times \log_2 \eta$
- Difficulty = $\frac{\eta_1}{2} \times \frac{N_2}{\eta_2}$



Tried classifiers

- Naive Bayes (NB)
- Bayesian networks (BayesNet)
- Decision trees (DecTree)
- Logistic regression (Logistic)



Results without new metrics

Terminology

$$PD = \text{Probability of detection} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \cdot 100\%$$

$$PF = \text{Probability of false alarms} = \frac{\text{false positives}}{\text{true negatives} + \text{false positives}} \cdot 100\%$$

$$Bal = \text{Balance} = 100 - \frac{\sqrt{(0 - PF)^2 + (100 - PD)^2}}{\sqrt{2}}$$

log : metrics at a logarithmic scale

note: the difference between the highest and the second highest value can be statistically not significant. See numbers in bold.



Results without new metrics

Linux

Classifier	PD	PF	Bal
NB	60.13	12.46	69.24
NB + log	92.33	37.09	72.28
BayesNet	3.72	1.85	31.91
BayesNet + log	5.34	1.95	33.06
DectTree	0.00	0.03	29.30
DectTree + log	0.00	0.00	29.30
Logistic	1.00	0.63	30.01
Logistic + log	3.84	0.62	32.01



Results without new metrics

MySQL

Classifier	PD	PF	Bal
NB	49.14	10.46	63.02
NB + log	88.30	34.67	73.53
BayesNet	44.57	8.65	60.26
BayesNet + log	44.50	8.65	60.21
DectTree	11.21	2.64	37.17
DectTree + log	11.28	2.65	37.22
Logistic	22.93	4.46	45.40
Logistic + log	25.07	5.13	46.88



Results without new metrics

CARDAMOM

Classifier	PD	PF	Bal
NB	0.00	7.18	29.08
NB + log	54.50	25.15	53.25
BayesNet	0.00	0.00	29.30
BayesNet + log	0.00	0.00	29.30
DectTree	0.00	0.00	29.30
DectTree + log	0.00	0.00	29.30
Logistic	0.00	1.21	29.30
Logistic + log	0.00	0.65	29.30



Results without new metrics

⇒ Select the Naive Bayes classifier with metrics at a logarithmic scale.

High probability of detection but high probability of false alarm. Still useful for Verification & Validation process:

avoids to inspect the files classified as non-ARB !



Attribute selection

Which attribute to select ?

For each project, sort metrics by *Information Gain*:

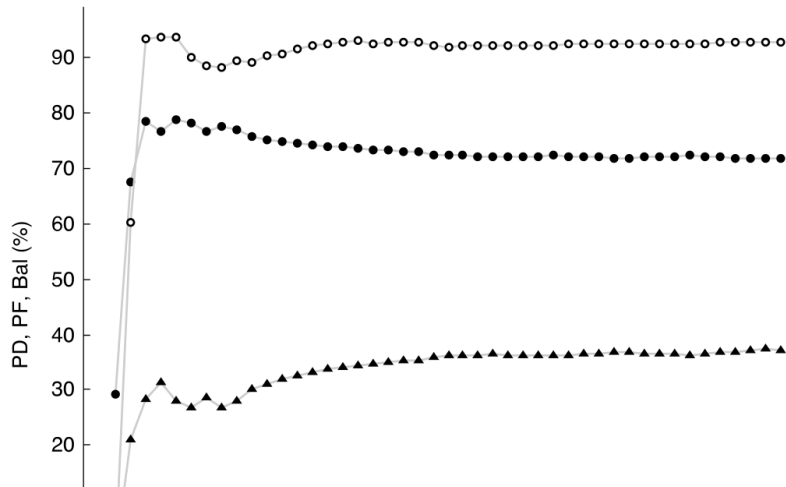
$InfoGain(A_i) =$ the number of bits of information obtained
by knowing the value of attribute A_i

Then, in a decreasing order of $InfoGain(A_i)$, compute PD, PF and Bal with the NB+log classifier.



Impact of attribute selection

Linux

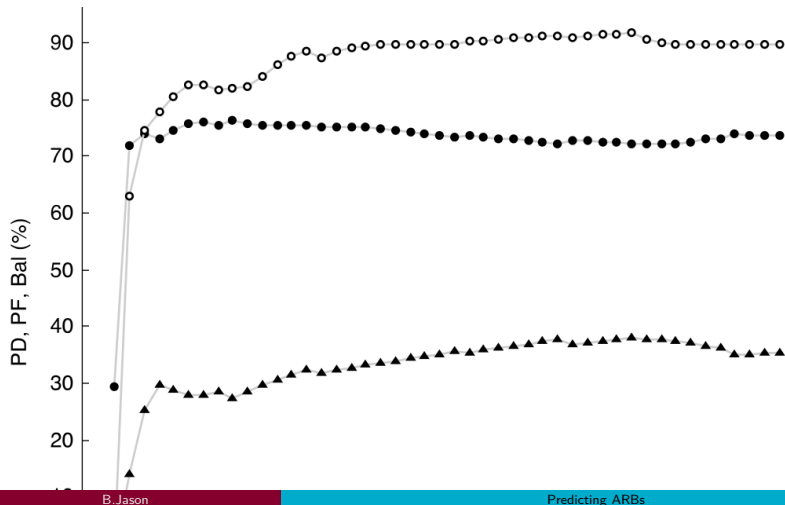


Top 10 metrics
AvgLineComment
AltAvgLineComment
AvgLine
AvgCyclomaticModified
SumCyclomaticStrict
Dif
CountLineComment
SumEssential
AltCountLineComment



Impact of attribute selection

MySQL

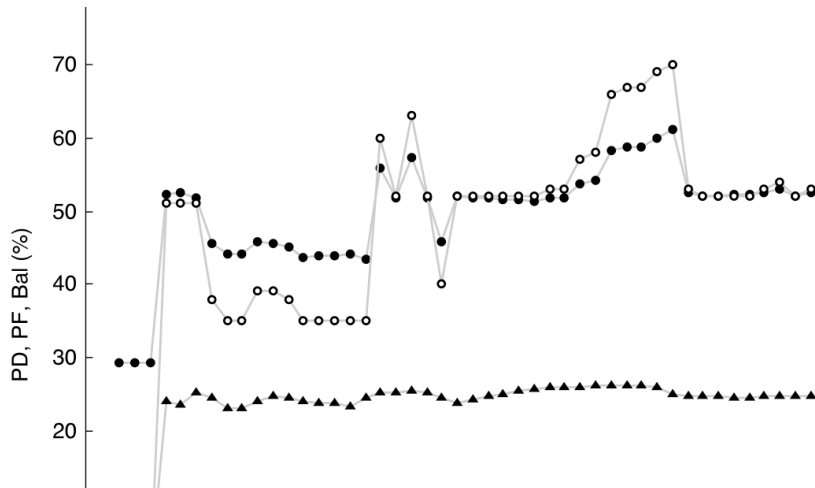


Top 10 metrics
CountLineBlank
CountLineCode
CountStmt
AltCountLineBlank
CountSemicolon
MaxCyclomaticModified
AltCountLineCode
CountStmtEmpty
MaxCyclomatic



Impact of attribute selection

CARDAMOM



Top 10 metrics
CountDeclFunction
CountLine
AvgLineCode
CountDeclClass
AvgLineComment
CountLineCodeExe
CountLineInactive
CountLineComment



Impact of attribute selection

- Top metrics not the same
- Need about 5 top metrics
- With all metrics, nearly the same results than with 5

⇒ We keep the full set of metrics.



New metrics

lot of ARB are about memory leaks.

6 new metrics related to memory usage, the **ARMs**:

- *AllocOps* : Number of instruction about allocation of memory
(malloc)
- *DeallocOps* : Number of instruction about Deallocation of memory
(free)



New metrics

- *DerefUse* : Number of times a pointer variable is dereferenced to write

Dereferencing a structure member

```
typedef struct X { int a; int b; } X;  
X x;  
X* p = &x;  
p->a = 3.14159;
```

- *DerefSet* : Number of times a pointer variable is dereferenced to read

Dereferencing p and read

```
printf("%d", p->a);
```

New metrics

- *UniqueDerefSet* : Same as DerefSet but variable counted one time per file
- *UniqueDerefUse* : Same as DerefUse but variable counted one time per file

Results with new metrics

Linux

Classifier	PD	PF	Bal
NB	60.13	12.46	69.24
NB + log	92.33	37.09	72.28
NB + ARMs	58.13	11.88	68.34
NB + log + ARMs	93.56	35.90	73.25

It decreases PF.

Results with new metrics

MySQL

Classifier	PD	PF	Bal
NB	49.14	10.46	63.02
NB + log	88.30	34.67	73.53
NB + ARMs	48.42	10.06	62.61
NB + log + ARMs	88.37	32.99	74.69

It decreases PF.



Results with new metrics

CARDAMOM

Classifier	PD	PF	Bal
NB	0.00	7.18	29.08
NB + log	54.50	25.15	53.25
NB + ARMs	0.00	6.57	29.12
NB + log + ARMs	68.00	10.22	70.32

It decreases PF and increases PD. \implies We keep ARMs.

Cross-component classification

In the same project, Can we predict ARB in a new component ?

Results of classifier when we exclude a component and use it as a test set:

Project	Component	PD	PF	Bal
Linux	Network drivers	88.9	41.7	69.5
	SCSI drivers	75.0	27.6	73.7
	EXT3	100.0	52.2	63.1
	IPv4	100.0	52.2	63.1
MySQL	InnoDB	65.6	16.5	73.0
	Replication	100.0	28.6	79.8
	Optimizer	100.0	69.7	50.7
CARDAMOM	Foundation	0.0	0.0	29.3
	Load Balancing	100.0	9.0	93.7

(0% of ARB detected in Foundation because there is only one ARB.)

Cross-project classification

Can those classifiers predict ARB for another project ?

Results of classifier when the training set comes from another project than the test set:

Train	Linux			MySQL			CARDAMOM		
	PD	PF	Bal	PD	PF	Bal	PD	PF	Bal
Linux	-	-	-	82.9	23.8	79.3	0.0	0.2	29.3
MySQL	100.0	50.7	64.2	-	-	-	33.3	11.8	52.1
CARDA	0.0	0.1	29.3	0.0	0.6	29.3	-	-	-

Low performance with CARDAMOM due to the difference of characteristics between CARDAMOM and the two others.

(CARDAMOM in pure C++, MySQL in mixed C/C++, Linux in pure C)

Conclusion

Predicting ARBs with software metrics is possible.

High PD: We can avoid inspecting files classified as non ARB-prone.

We have to check if characteristics are similar before using a classifier for another project.

What's International Trade About?

- Do you have any expectation for this course?
- 迄今，你认为 2018 年经济领域发生的最 significant 的事是什么？
- 如果你想赢，你必须先知道它是如何运作的

2017 年中国贸易情况

2017 年，中国**货物贸易总额 27.79 万亿人民币**，折合 **4.105 万亿美元**。其中，出口 15.33 万亿元，增长 10.8%；进口 12.46 万亿美元，增长 20.9%；贸易顺差 2.87 万亿美元，收窄 14.2%；

2017 年，中国前三大贸易伙伴分别是**欧盟、美国和东盟**，三者合计占我国进出口总值的 41.8%；

2017 年，中国**一般贸易**进出口 15.66 万亿元，占比 56.4%，贸易结构有所优化；

2017 年，中国出口最多的产品是**机电类产品**，共出口 8.95 万亿，占比 58.4%，传统劳动密集型产品合计出口 3.08 万亿元，占比 21.1%；进口最多的三类商品是原油、铁矿石，汽车。