The 39th International Conference on Software Engineering

What Causes My Test Alarm? Automatic Cause Analysis for Test Alarms in System and Integration Testing

Authors: He Jiang[1], **Xiaochen Li**[1], Zijiang Yang[2], Jifeng Xuan[3]
[1]Dalian University of Technology, [2]Western Michigan University, [3]Wuhan University

海纳百川 自强不息 厚德笃学 知行合一

Background

System and integration testing (SIT)

- Continuous integration increases SIT's frequency .
  - DevOps: faster time to market
  - Cloud-based system: run 1,000 test scripts in 25 minutes

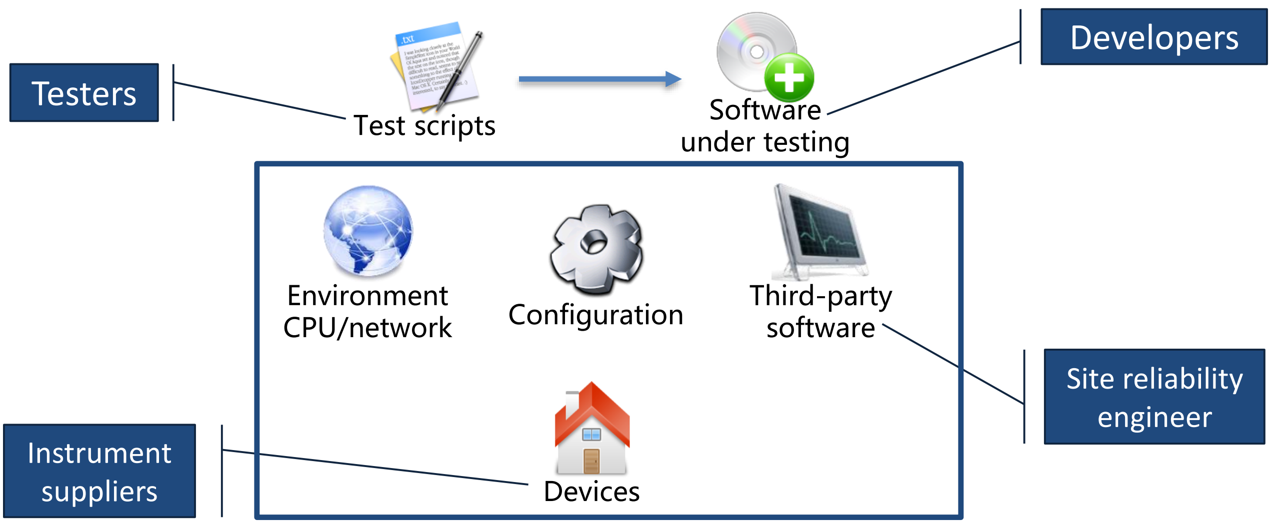
- Running test scripts in SIT may fail.
  - We find 6000+ failures in a single month in one product

- Testers need to figure out the failure causes
  - Require the stakeholders to fix them

Background

Test software in SIT

- To test software
  - Many artifacts and stakeholders are involved
  - Any artifact may have defects

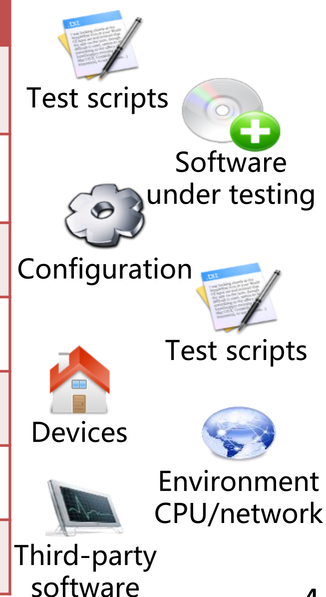


Background

Test alarms in SIT

- Test scripts may fail for various causes
  - A test alarm is an alarm to warn the test script failure

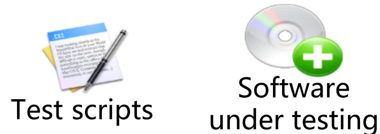
ID	Type of cause	Testers' solution
C1	Obsolete test	update test scripts
C2	Product code defect	submit bugs to developers
C3	Configuration error	correct configuration files
C4	Test script defect	debug test scripts
C5	Device anomaly	submit bugs to instrument suppliers
C6	Environment issue	diagnose the environment
C7	Software problem	ask site reliability engineers to diagnose



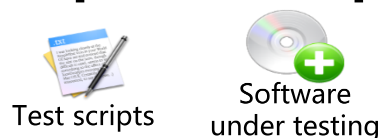
Related Work

Classify test alarms (academic)

- Product code defect or Test script defect [Rogstad et al. 15]
  - For database applications



- Product code defect or Obsolete test [Hao et. al. 13]
  - Unit testing
  - First decision tree



- Product code defect or others [Herzig & Nagappan 15]
  - Association rules / Binary Classification

REF:

1. E. Rogstad, and L. C. Briand, Clustering deviations for black box regression testing of database applications. IEEE Trans. on Reliability
2. D. Hao, T. Lan, H. Zhang, C. Guo, and L. Zhang. Is this a bug or an obsolete test? In ECOOP
3. K. Herzig and N. Nagappan. Empirically detecting false test alarms using association rules. ICSE, 2015

Related Work

Classify test alarms (academic)

- Product code defect or Test script defect [Rogstad et al. 15]
  - For database applications

Test scripts

Software under testing

- Product code defect or Obsolete test [Hao et. al. 13]
  - Unit testing
  - First decision tree

- Product code defect or othe
  - Association rules / Binary C

The causes are more complex than binary classification

Test scripts

REF:

1. E. Rogstad, and L. C. Briand, Clustering deviations for black box regression testing of database applications. IEEE Trans. on Reliability
2. D. Hao, T. Lan, H. Zhang, C. Guo, and L. Zhang. Is this a bug or an obsolete test? In ECOOP
3. K. Herzig and N. Nagappan. Empirically detecting false test alarms using association rules. ICSE, 2015

Related Work

Classify test alarms (industry)

- A survey to industry testers
  - They collect test logs of failed test scripts
  - They manually build regular expressions for classification
  - Accuracy is 20%-30% over distinct projects

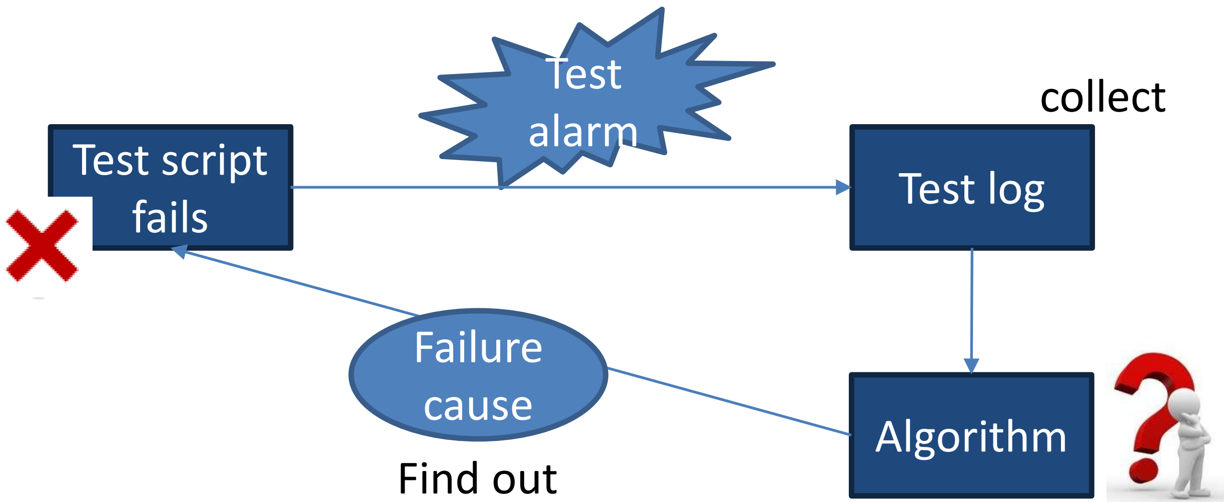
#	Regular expression	Cause type	Severity
1	topomatch fail	Environment issue	normal
2	Info: GEN_ERROR_FILE_OPEN	Environment issue	normal
3	Error: The current mode is unframed mode. Please delete it first	Test script defect	normal
4	Error: Operation abnormal	Product code defect	severe

The Problem

Test alarm analysis

- Analyze the cause of test alarms
  - Test logs are easy to get
  - Testers also read test logs to analyze the alarms

Classification before bug location, bug fixing etc.



The Problem

A test log

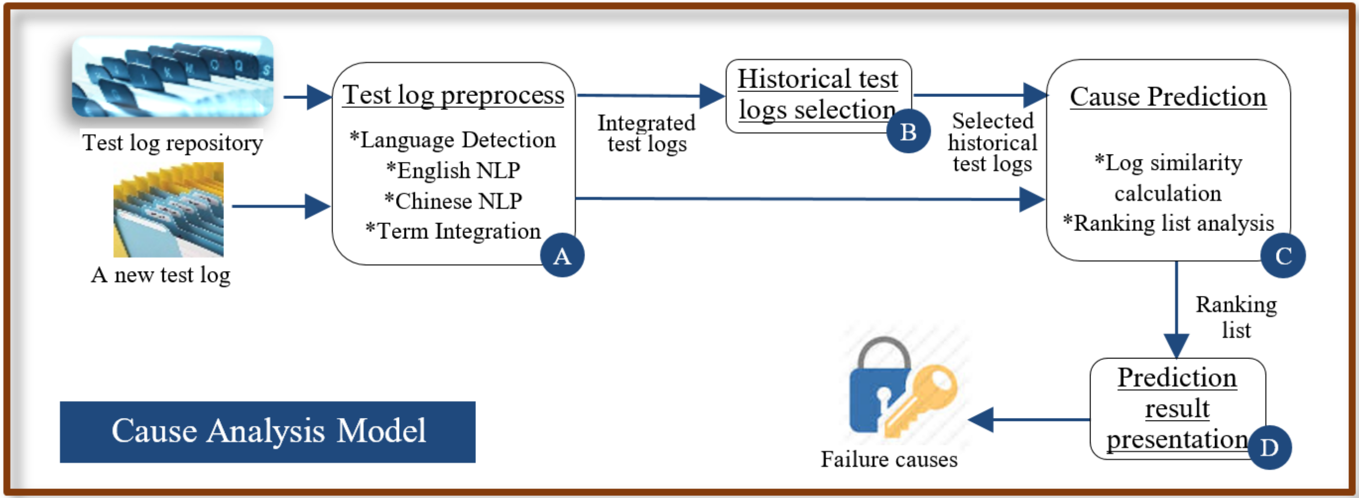
- Bilingual documents: English & Chinese
- Long: more than 1000 lines, more than 10GB (14,000 logs)

```
[2015-06-03 02:39:24.687]【189.106.7.11】[189.106.7.12 23]: cd /opt/VNFP/0

cd /opt/VNFP/0

-bash: cd: /opt/VNFP/0: No such file or directory
imageVMNPSO-001:~ #

[SSP_INFO] >>>>>>>>>>>>>>>>>>>>rs cd /opt/VNFP/0
-bash: cd: /opt/VNFP/0: No such file or directory
imageVMNPSO-001:~ #  -#<D:/CIEnv_Fenix_HY/Test_Suit/IGP_DevM/SPC/common/Fenix/Feni

在测试步骤《查询虚拟机信息》中的检查点《获得vappid 成功》里出现断言失败
出错文件： C:/Program Files/Impeller/lib/ruby/lib/ruby/gems/1.8/gems/testlib-VrpBas
错误信息： 登录指定的文件夹失败
期望值：false
实际值：true
调用函数： assert_false
相关文件： D:/CIEnv_Fenix_HY/Test_Suit/IGP_DevM/SPC/common/Fenix/Fenix_mano.rb:373
```

Cause Analysis Model (CAM)

Framework

- CAM's Idea
  - Search the test logs of historical test alarms that may have the same failure cause with the new test log



Cause Analysis Model (CAM)

An example snippet

- A test log snippet of function point "AUTO UPDATE SCHEMA" (AUS)
  - Each test script is associated with a func. point
  - Func. points are functional requirements for the software
  - A test script verifying function "configure network proxy" may add "**NETCONF_PROXY_FUNC**" as the func. point

New test log snippet with function point "AUTO UPDATE SCHEMA (AUS)"
E [**exception happens continuously for more than 20 times**] [2015-06-28 02:10:52.964] timed out while waiting for more data

Cause Analysis Model (CAM)

Test log preprocess

- Language Detection

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [**exception happens continuously for more than 20 times**]
[2015-06-28 02:10:52.964] timed out while waiting for more data

Cause Analysis Model (CAM)

Test log preprocess

- Language Detection
- English NLP
  - Tokenization,
  - Stop words removal

  (single letters, punctuation marks, and numbers),
  - Stemming

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [**exception happens continuously for more than 20 times**]
[2015-06-28 02:10:52.964] timed out while waiting for more data

~~E [] [2015-06-28 02:10:52.964]~~ \ time~~d~~ \ ~~out~~ \ while \ wait~~ing~~ \ ~~for \\~~ more \ data

Cause Analysis Model (CAM)

Test log preprocess

- Language Detection
- English NLP
  - Tokenization,
  - Stop words removal

    (single letters, punctuation marks, and numbers),
  - Stemming
- Chinese NLP
  - Word segmentation

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [**exception happens continuously for more than 20 times**]
[2015-06-28 02:10:52.964] timed out while waiting for more data

~~E [] [2015-06-28 02:10:52.964]~~ \ time~~d~~ \ ~~out~~ \ while \ wait~~ing~~ \ ~~for~~ \ more \ data

exception \ happens \ continuously \ for more than \ ~~20~~ \times

Cause Analysis Model (CAM)

Test log preprocess

- Language Detection
- English NLP
  - Tokenization,
  - Stop words removal

  (single letters, punctuation marks, and numbers),
  - Stemming
- Chinese NLP
  - Word segmentation
- Term Integration

New test log snippet with function point “AUTO UPDATE SCHEMA (AUS)”
E [**exception happens continuously for more than 20 times**]
[2015-06-28 02:10:52.964] timed out while waiting for more data

~~E [] [2015-06-28 02:10:52.964]~~ \ time~~d~~ \ ~~out~~ \ while \ wait~~ing~~ \ ~~for~~ \ more \ data

exception \ happens \ continuously \ for more than \ ~~20~~ \times

**exception \ happens \ continuously \ for more than \ times **
time \ while \ wait \ more \ data

Cause Analysis Model (CAM)

Historical test log selection

- Select historical test logs by func. point
  - Select all, if no matched func. point

New test log snippet with function point "AUTO UPDATE SCHEMA" (**AUS**)
E [**exception happens continuously for more than 20 times**] [2015-06-28 02:10:52.964] timed out while waiting for more data

Logs	Func. Point	Cause
his3	AUS	C2
his4	AUS	C3
his1	AUS	C3
his2	AUS	C3
his5	AUS	C2
his6	NPF	C1
his7	NPF	C3

Cause Analysis Model (CAM)

Cause prediction

- Log similarity with selected logs
  - 2-shingling terms (successfully applied in information retrieval)
  - TF-IDF based cosine similarity

**exception \ happens \ continuously \ for more than \ times ** time \ while \ wait \ more \ data

**exception happens **
**happens continuously **
**continuously for more than **
**for more than times **
times time \
time while \
while wait \
wait more \
more data

Logs	Func. Point	Sim_{log}	Cause
his3	AUS	0.586	C2
his4	AUS	0.472	C3
his1	AUS	0.322	C3
his2	AUS	0.320	C3
his5	AUS	0.134	C2

Cause Analysis Model (CAM)

Cause prediction

- Predict by k-Nearest Neighbor
  - Case 1: the similarity of top 1 log (his3) exceeds a threshold
  - Case 2: the similarity of top 1 log (his3) is lower than a threshold
    - C2=0.586+0.134; **C3**=0.472+0.311+0.320

Case 1 threshold=0.5

Logs	Func. Point	Sim_{log}	Cause
his3	AUS	0.586	**C2**
his4	AUS	0.472	C3
his1	AUS	0.322	C3
his2	AUS	0.320	C3
his5	AUS	0.134	C2

Case 2 threshold=0.6

Logs	Func. Point	Sim_{log}	Cause
his3	AUS	0.586	C2
his4	AUS	0.472	C3
his1	AUS	0.322	C3
his2	AUS	0.320	C3
his5	AUS	0.134	C2

Cause Analyze Model (CAM)

Prediction result presentation

- Present differences between the new log and the most similar test log of the same cause
  - ➢ Testers are familiar with historical test logs
  - ➢ Comparison may be more easier

***new* test log**	***historical* test log**
cd/opt/VNFP/0 -bash: cd imageVMNPSO-001 assertion fails	rm /opt/VNFP/0 imageVMNPSO-001 assertion fails
C1	C1

Experimental Setup

Dataset

- Two industrial testing projects at Huawei-Tech Inc.
- Logs about one month per project
- More than 14,000 test logs
- Focus on
  one failure cause
  per test log

#	Dataset / Info	DS1		DS2	
1	# Test logs	7663		6977	
2	Size	4.72GB		6.06GB	
3	Time Frame	June 1st – July 30th, 2015		Oct. 26th – Nov. 16th, 2015	
4	# Testing day	40 day		22 day	
5	# Test logs per day	192		317	
6	# Avg. lines	942 lines		1375 lines	
7	# Avg. test steps	247 test steps		344 test steps	
8	# Obsolete test (C1)	1185	15.46%	*	*
9	# Product code defect (C2)	4459	58.19%	1963	28.14%
10	# Configuration error (C3)	761	9.93%	345	4.94%
11	# Test script defect (C4)	892	11.64%	3259	46.71%
12	# Device anomaly (C5)	335	4.37%	298	4.27%
13	# Environment issue (C6)	19	0.28%	168	2.41%
14	# Software problem (C7)	12	0.17%	944	13.53%
15	# Avg. type of causes per day	3.85 per day		3.86 per day	

Experimental Setup

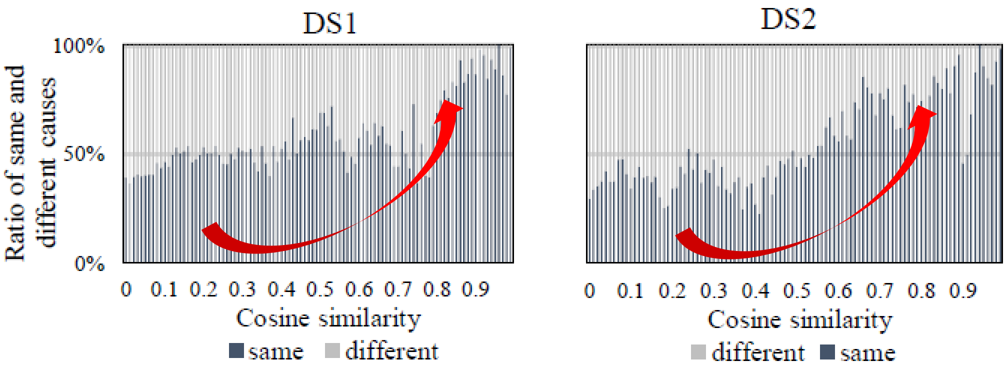
- Evaluation method
  - Accuracy、Area-Under-Curve
  - Running time, memory consumption
  - Incremental framework (simulate testers' daily work)

- Baseline Algorithms: bag-of-words
  - Lazy Associative Classifier (LAC)
  - Best First Tree (BFT).
  - Topic Model (TM)

Experimental Results

Evaluate CAM's hypothesis

- Are the test logs with the same causes more similar than those with different causes ?



  - ➢ As the similarity grows, more and more test logs are in the same failure cause
  - ➢ Test logs with the same causes are more similar

Experimental Results

Overall performance

- How does CAM perform against baseline algorithms?

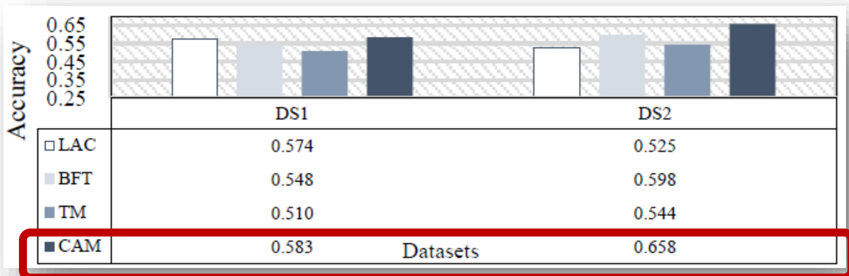


Fig. 1 Accuracy for algorithms on two datasets

➢ Outperform the baseline algorithms ($p<0.05$)

Experimental Results

Overall performance

- How does CAM perform against baseline algorithms?

Algorithm \ Cause		C1	C2	C3	C4	C5	C6	C7
DS1	LAC	0.61	0.57	0.48	0.52	0.50	0.33	0.51
	BFT	0.73	0.65	**0.66**	0.60	**0.77**	0.40	**0.70**
	TM	0.68	0.67	0.56	0.58	0.62	0.50	0.54
	CAM	**0.77**	**0.71**	0.59	**0.61**	0.62	0.50	0.62
DS2	LAC	-	0.60	0.53	0.64	**0.63**	**0.83**	0.73
	BFT	-	0.67	0.65	0.70	0.60	0.77	0.86
	TM	-	0.62	0.51	0.68	0.52	0.77	0.78
	CAM	-	**0.68**	**0.66**	**0.81**	0.51	0.74	**0.87**

Fig. 2 Comparison on AUC

- Outperform the baseline algorithms ($p<0.05$)
- Superior over the majority of cause types

Experimental Results

Overall performance

- How does CAM perform against baseline algorithms?

Algorithm	Time (in minutes)						Memory	
	DS1 (7356 test logs)			DS2 (6557 test logs)			DS1	DS2
	Training	Test	Total	Training	Test	Total		
LAC	11.4	1	12.4	3.6	1.4	5	3 GB	3 GB
BFT	208.6	0.3	208.9	46.8	0.2	47	22 GB	20 GB
TM	75.1	2.8	77.9	142	4.3	146.3	8 GB	5 GB
CAM	0	6.9	6.9	0	14.4	14.4	4 GB	4 GB

Fig. 3 Comparison on computation resources consumption

- Outperform the baseline algorithms ($p<0.05$)
- Superior over the majority of cause types
- Resources saving, take about 0.1s and less than 4GB memory to process a test log.

Experimental Results

Historical test log selection

- How does historical test log selection work?
  - CAM-FP: CAM without historical test log selection

Algorithm	DS1			DS2		
	Accuracy	Total time	Memory	Accuracy	Total time	Memory
CAM-FP	0.555	39.2 min	4GB	0.634	46.4 min	4GB
CAM	0.583	6.9 min	4GB	0.658	14.4 min	4GB

Fig. 4 Accuracy, total time, and memory for CAM and CAM-FP

  - Selection reduces noisy and shortens running time

Experimental Results

Historical test log selection

- How does historical test log selection work?
  - CAM-FP: CAM without historical test log selection

Algorithm	DS1			DS2		
	Accuracy	Tota				
CAM-FP	0.555	39.				
CAM	0.583	6.9				

Fig. 4 Accura
for

Algorithm \ Cause		C1	C2	C3	C4	C5	C6	C7
DS1	CAM-FP	0.73	0.70	0.59	0.57	0.59	0.50	0.62
	CAM	**0.77**	**0.71**	0.59	**0.61**	**0.62**	0.50	0.62
DS2	CAM-FP	-	0.67	**0.76**	0.76	**0.52**	0.67	0.84
	CAM	-	**0.68**	0.66	**0.81**	0.51	**0.74**	**0.87**

Fig. 5 AUC values for CAM and CAM-FP

  - Selection reduces noisy and shortens running time
  - Without selection, CAM-FP still achieves competitive performance

Experimental Results

Evaluation in real scenario

- How does CAM perform in a real development scenario?
  - 72% accuracy after running for two months.

- Feedback
  - CAM is better than manually building regular expressions.
  - Actually, I will not believe in an automatic tool. However, after presenting the historical test logs, I can quickly decide whether the prediction is correct. **CAM accelerates my work**.
  - **Suggestions**: labeling the defect-related snippets, provide suggestions on how to fix defects

Conclusion

In this paper, we

- Propose a new approach to address automatically analyzing the test alarm causes in SIT.
- Construct two industrial datasets [http://oscar-lab.org/cam/].
  The failure causes are manually labeled and verified by testers.
- Conduct a series of experiments to investigate CAM.
  CAM is both effective and efficient.
- Deploy and evaluate CAM in a real development scenario.

Thanks

What Causes My Test Alarm? Automatic Cause Analysis for Test Alarms in System and Integration Testing

Reporter： Xiaochen Li

Dalian University of Technology, China

Authors: He Jiang[1], Xiaochen Li[1], Zijiang Yang[2], Jifeng Xuan[3]
[1]Dalian University of Technology, [2]Western Michigan University, [3]Wuhan University