A survey of root cause analysis

霍茵桐

2021/06/25 Workshop

HUAWEI

ARISE
Automated Reliable Intelligent Software Engineering

香港中文大學
The Chinese University of Hong Kong

Problem Formulation

GOAL

Target to identify the underlying causes leading to a failure that has affected end users, which is often closely related to the failure diagnosis.

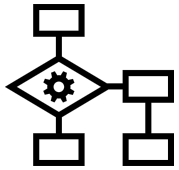
SOURCE

In the broad domain of root cause analysis, log-based failure diagnosis is now a standard practice.

CHALLENGES

- The complexity of modern software systems grows rapidly, making too complex to efficiently disclose relations of fault, failure and human-observed symptoms.
- As software systems become more mature, failures are becoming more and more hard to detect and diagnose.

In the very beginning…



Traditional rule-based methods are adopted.

- which heavily rely on a set of predefined rules (if-then logic) from the expert knowledge to diagnose failures.

Limitations:

Rule-based methods cannot be well generalized to unseen failures that are not included in the rules.

Outline

#1: Execution Replay Methods

#2: Model-based Methods

#3: Statistics-based Methods

#4: Retrieval-based Methods

Outline

#1: Execution Replay Methods

- Infer execution flow
- Trace back to software failure

#2: Model-based Methods

#3: Statistics-based Methods

#4: Retrieval-based Methods

Execution Replay Methods

Approach

Automatically infer the execution flow from logs and trace back to the software system failure.

Input:

- Log recorded during the failed execution
- Source code of the target program

Output:

- Derive the potential code paths and execution path

Execution Replay Methods (1)

LogMap Approach

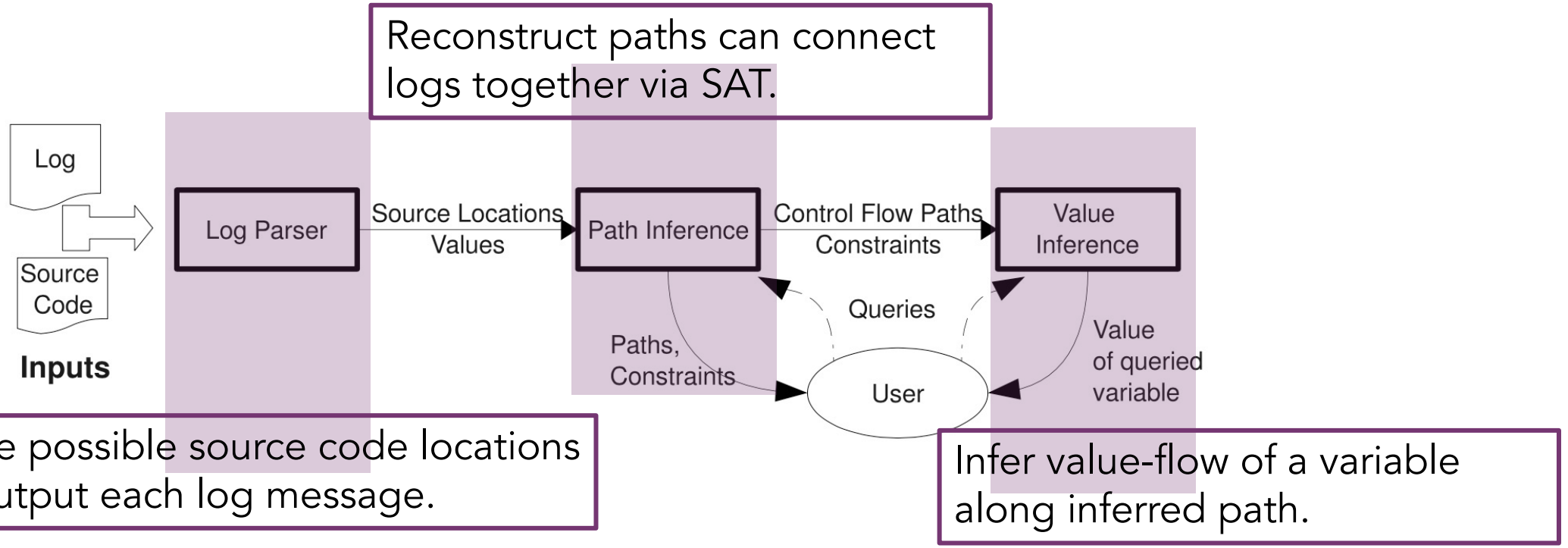
1. LogMap *retrieves log messages* from bug reports.
2. Apply static analysis technique to *identify corresponding logging lines* in source code.
3. Traverse through logging lines to *derive the potential code paths*, which help reconstruct the execution path and assist debugging process.

[1] Chen A R. An empirical study on leveraging logs for debugging production failures[C]//2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). IEEE, 2019: 126-128.

Execution Replay Methods (2)

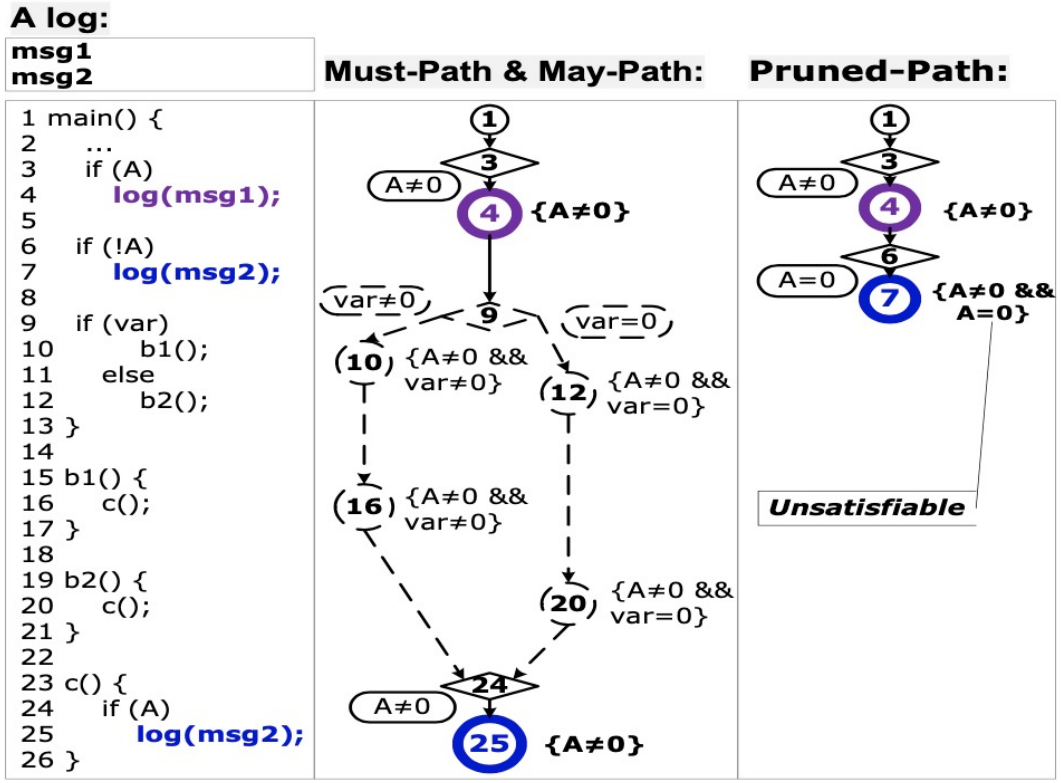
Goal: Find *information* from printed log

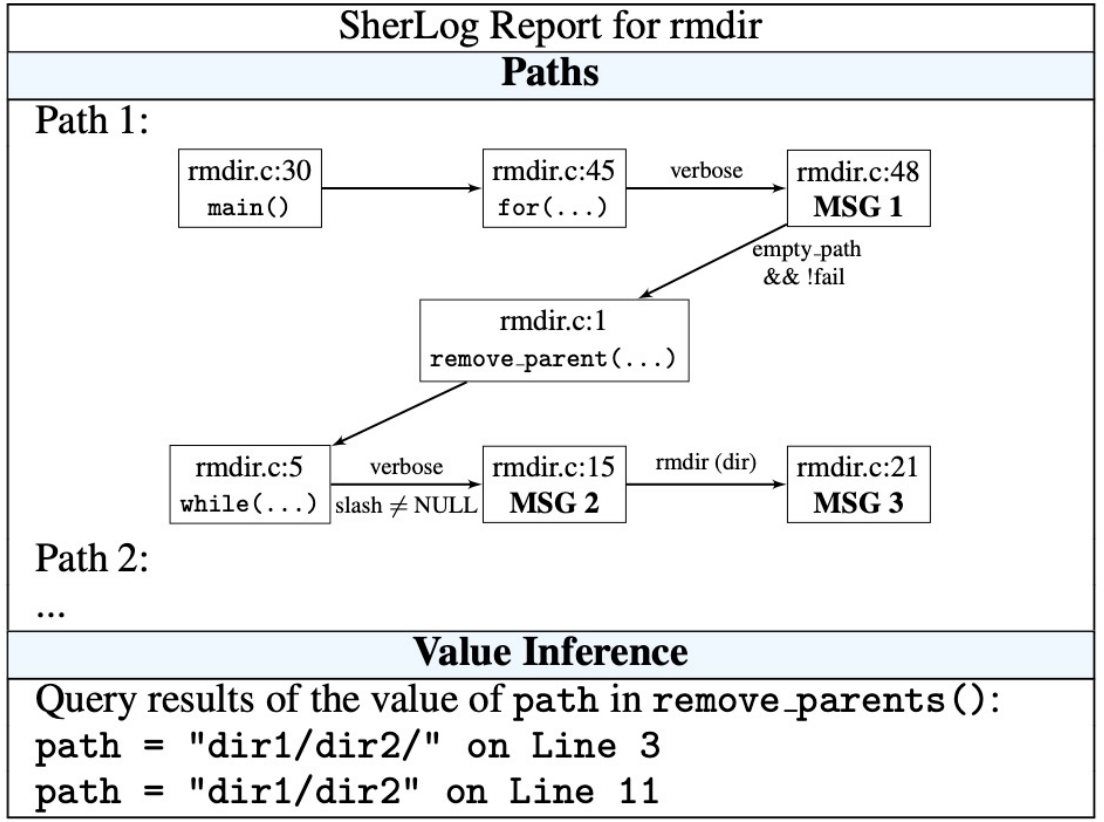
- *Must-have: P*artial execution paths that were definitely executed.
- *May-Have: P*artial execution paths that may have been executed
- *Must-Not-Have*: Execution paths that were definitely NOT executed.



[1] Yuan D, Mai H, Xiong W, et al. Sherlog: error diagnosis by connecting clues from run-time logs[C]//Proceedings of the fifteenth International Conference on Architectural support for programming languages and operating systems. 2010: 143-154.

Execution Replay Methods (2)





Outline

#1: Execution Replay Methods

#2: Model-based Methods

- Build reference model
- Check violate

#3: Statistics-based Methods

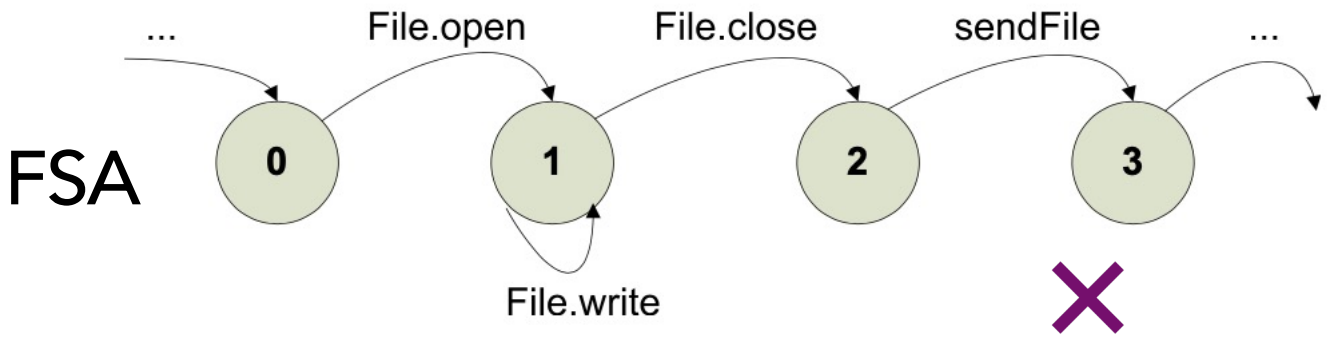
#4: Retrieval-based Methods

Model-based Methods

Approach

Utilize logs to build the reference model for a software system, then check which log events violate the reference model.

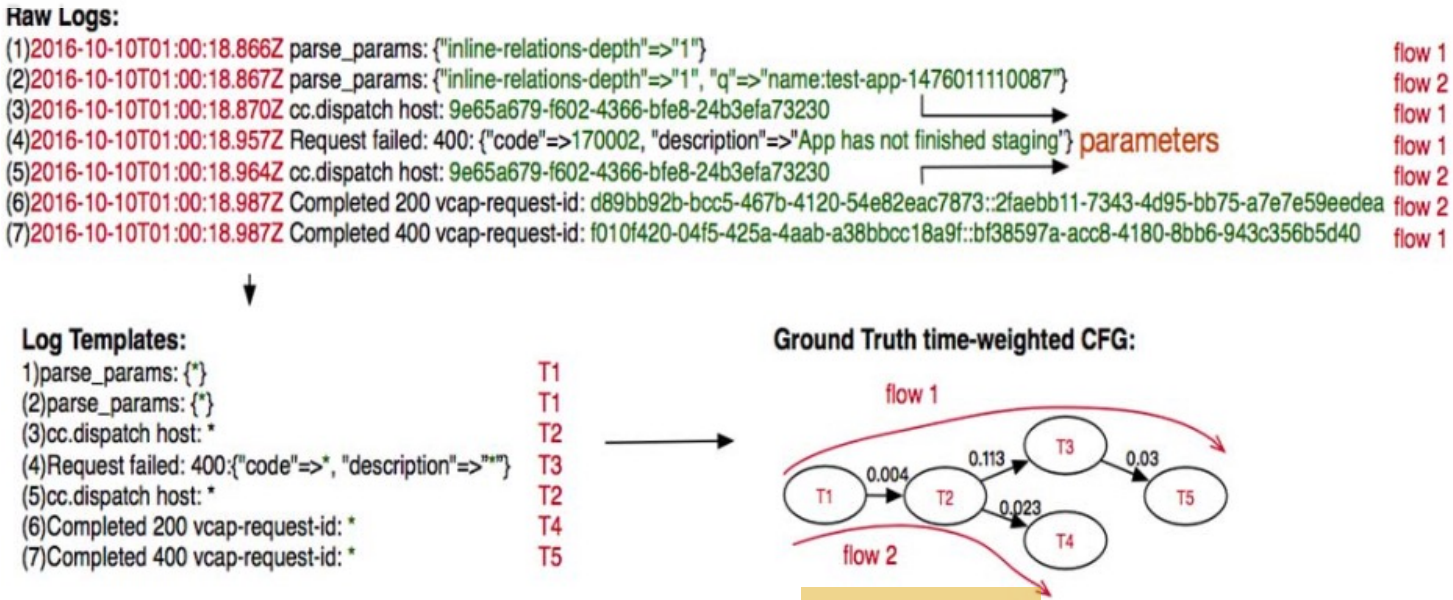
1. *Generate a model* of the legal behavior of a target system by tracing successful executions at testing-time.
2. *Compare* the set of events detected during failing executions with the generated models.



File.open, File.write, . . . , File.write, sendFile

Model-based Methods (1)

- How to generate reference model from log?
- How to compare?



Mine time-weighted control flow graphs (TCFG) from interleaved logs of a system.

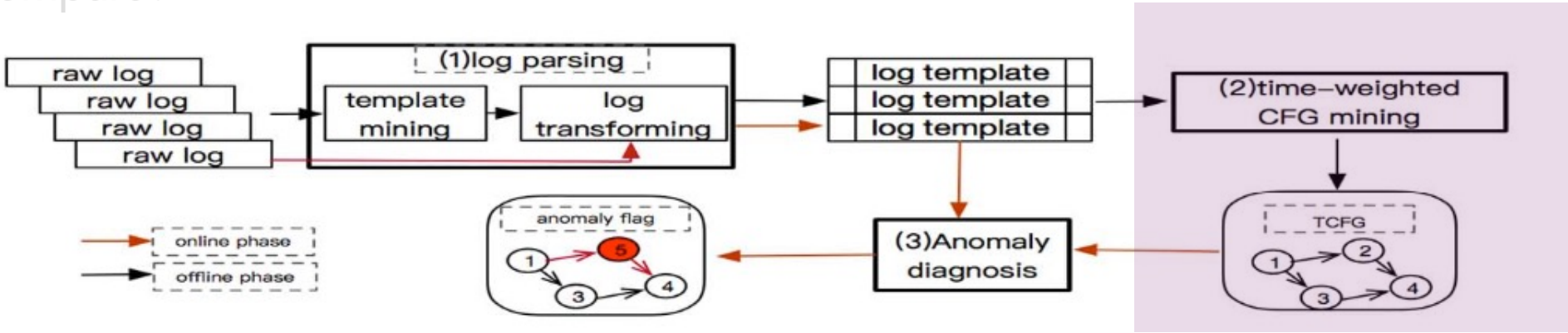
Node: *Template*
Edge: *Transition from one template to another*
Weighted: *Execution time between templates*

Jia T, Yang L, Chen P, et al. Logsed: Anomaly diagnosis through mining time-weighted control flow graph in logs[C]//2017 IEEE 10th International Conference on Cloud Computing (CLOUD). IEEE, 2017: 447-455.

Model-based Methods (1)

How to generate reference model from log?

How to compare?



Challenges: Without a certain transaction ID.

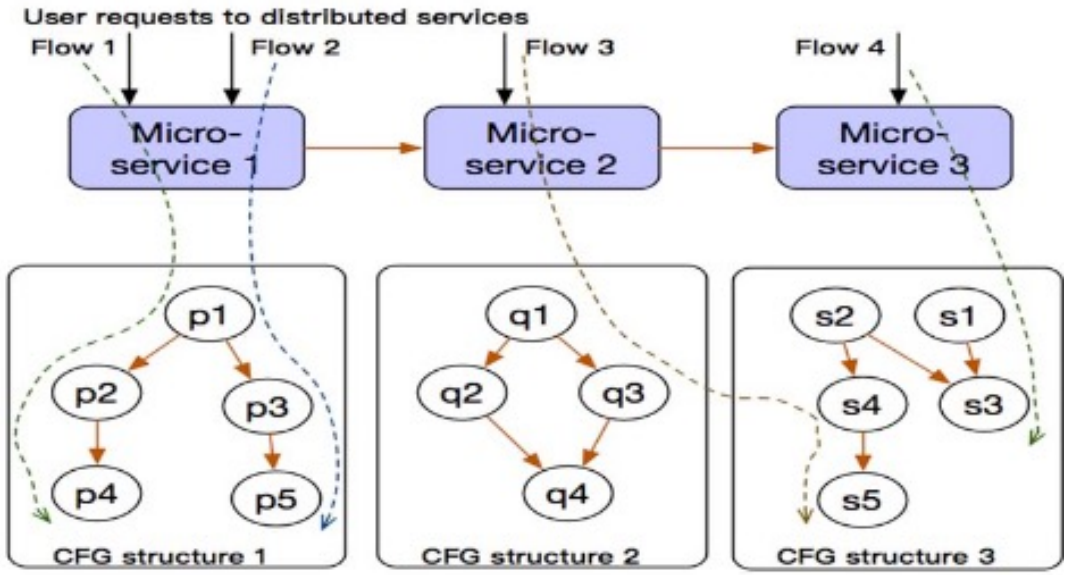
Motivation: Logs of correct immediately succeeding template will appear in a short period.

Two Stages:
(1) Compute frequent successor groups called FS groups of each template.
(2) Leverage this FS group to mine only the immediate successors of each template.

Jia T, Yang L, Chen P, et al. Logsed: Anomaly diagnosis through mining time-weighted control flow graph in logs[C]//2017 IEEE 10th International Conference on Cloud Computing (CLOUD). IEEE, 2017: 447-455.

Model-based Methods (2)

- How to generate reference model from log?
- How to compare?



Model the causal relationships between services as a service topology.

Node: *Service*
Edge: *Causal relationship between two services*

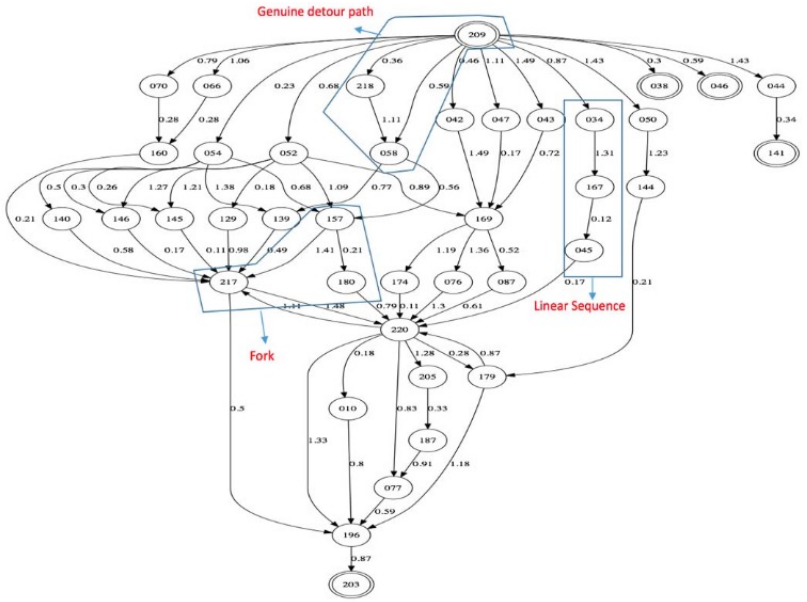
Mining approach: PC-algorithm

Jia T, Chen P, Yang L, et al. An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services[C]//2017 IEEE International Conference on Web Services (ICWS). IEEE, 2017: 25-32.

Model-based Methods

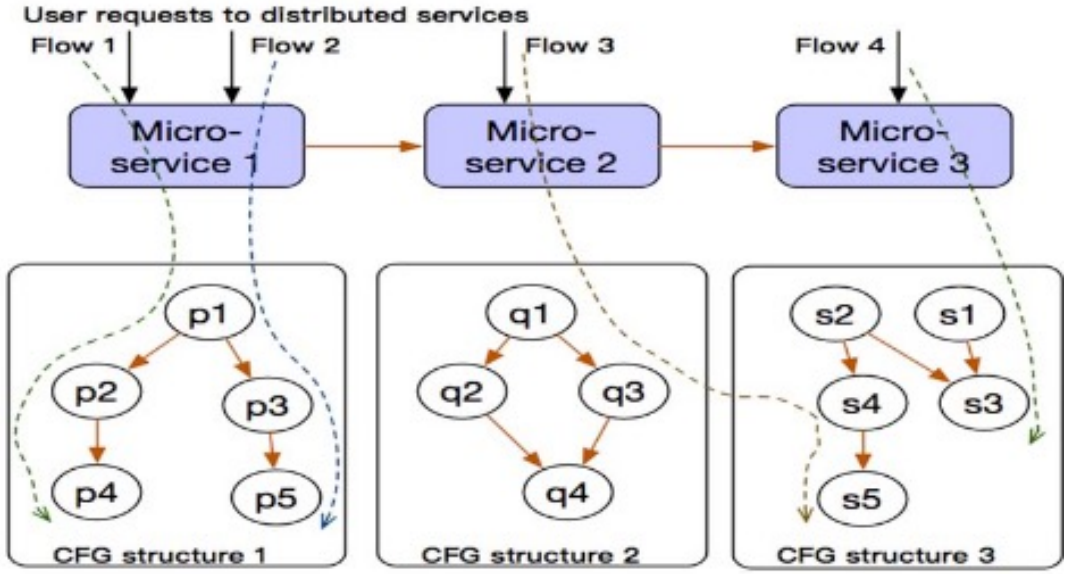
How to generate reference model from log?

How to compare?



TCFG

Diagnose for sequence anomaly, redundancy anomaly and latency anomaly.



Service topology.

Examine the TCFGs of its predecessor services in the service topology.

Model-based Methods

How to generate reference model from log?

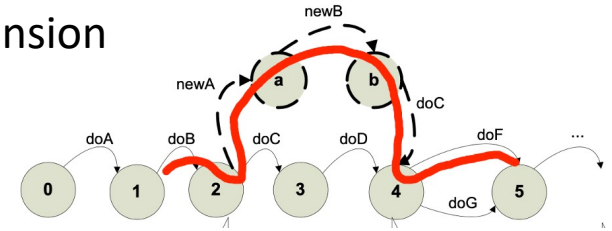
How to compare?

(delete, insert, replace and final state)

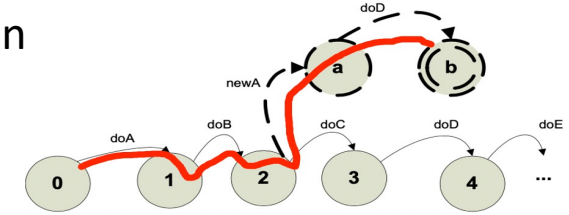
1 Select the expected behaviors to be considered for local analysis.

2 Compare expected behavior with the observed failed sequence.

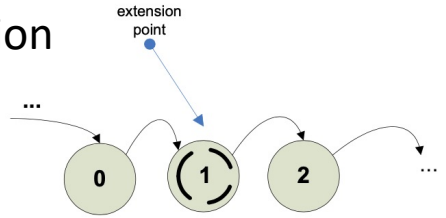
Branch extension



Tail extension



Final state extension



Example insert interpretation:
expected sequence: **doA doB - - - doC**
observed sequence: **doA doE doF doG doH doC**

Example delete interpretation:
expected sequence: **dol doL doF doG - doM**
observed sequence: **dol - - - doN doM**

Sets specific to anticipation
I.newEvents = doE doF doG doH
D.removedEvents = doL doF doG

Aligned sequences:
doE doF doG doH
doL doF doG -

alig = doF doG
notAlig.n = doL
notAlig.g = doH

General sets
I.m = doA doC
D.m = dol doM
I.d = /
D.a = doN

$$\text{score} = \frac{2*2-2*1-1+2+2-0-1}{6+6} = 0.33$$

confidence value =(0.33+1)/2=0.665
(normalized score)

no *long* expected behaviors considered in the analysis

Babenko A, Mariani L, Pastore F. Ava: Automated interpretation of dynamically detected anomalies[C]//Proceedings of the eighteenth international symposium on Software testing and analysis. 2009: 237-248.

Outline

#1: Execution Replay Methods

#2: Model-based Methods

#3: Statistics-based Methods

- Statistical analysis

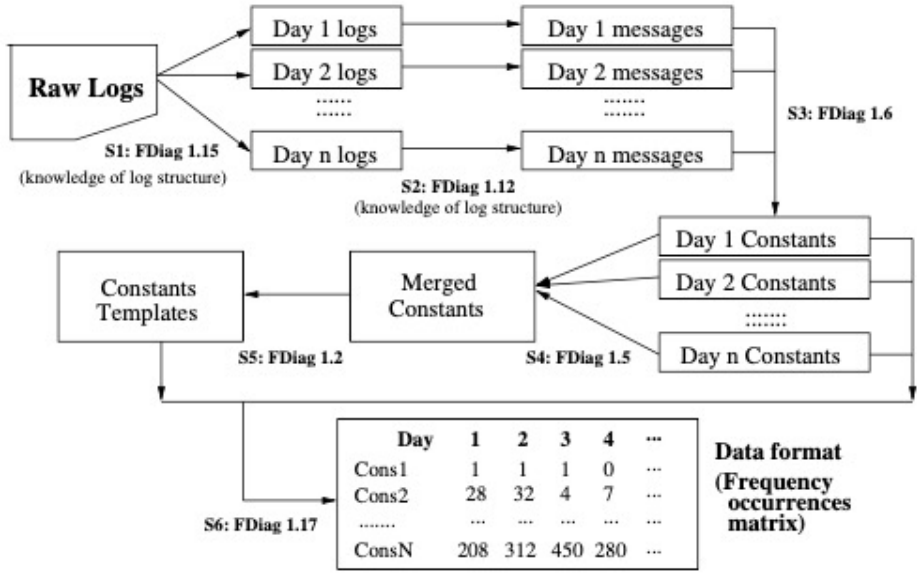
#4: Retrieval-based Methods

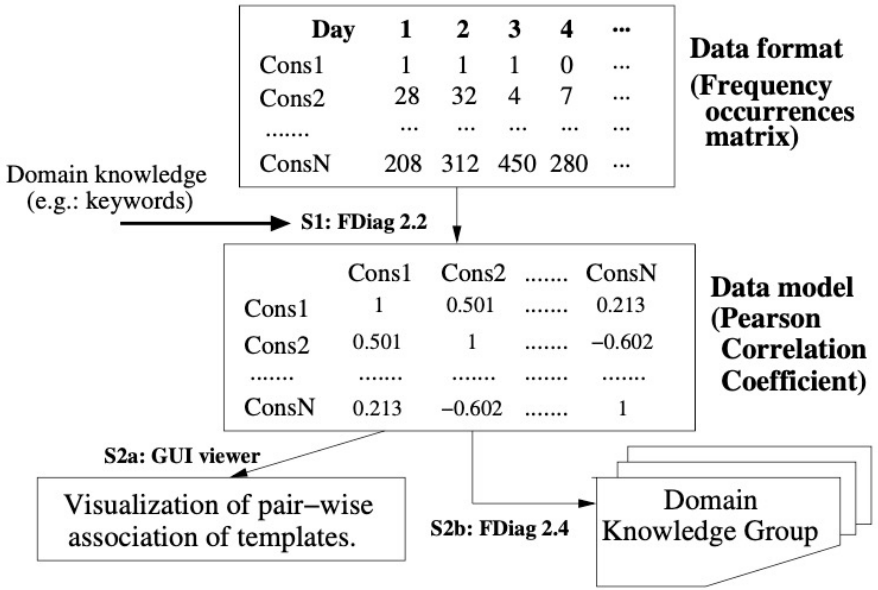
Statistics-based Methods

Approach

Employ some statistical techniques to capture the relationships between logs and the consequent failures.

Given knowledge of any system event, find all the system events that are associated with the given event, and how strongly associated these system events are to the given event.





Chuah E, Kuo S, Hiew P, et al. Diagnosing the root-causes of failures from cluster log files[C]//2010 International Conference on High Performance Computing. IEEE, 2010: 1-10.

Outline

#1: Execution Replay Methods

#2: Model-based Methods

#3: Statistics-based Methods

#4: Retrieval-based Methods

- Information Retrieval

Retrieval-based Methods

Approach

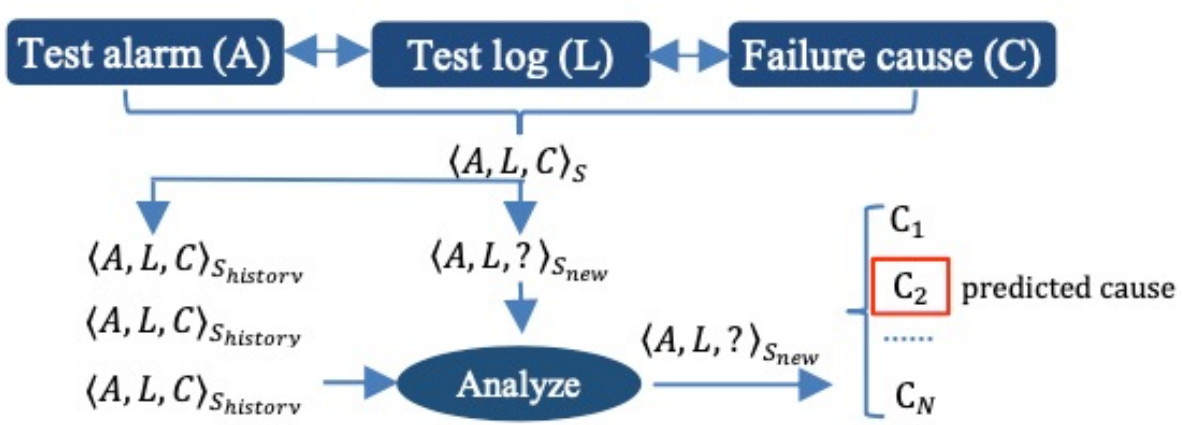
Retrieve similar failures in a knowledge base composed of failures in history.

Motivation: In practice, failures that previously occurred are valuale since they can aid developers in better diagnosing newly-occurred failures

Retrieval-based Methods

Basic Idea: Detect the test logs of historical test alarms that may share the same causes with the new test logs.

Bug Report

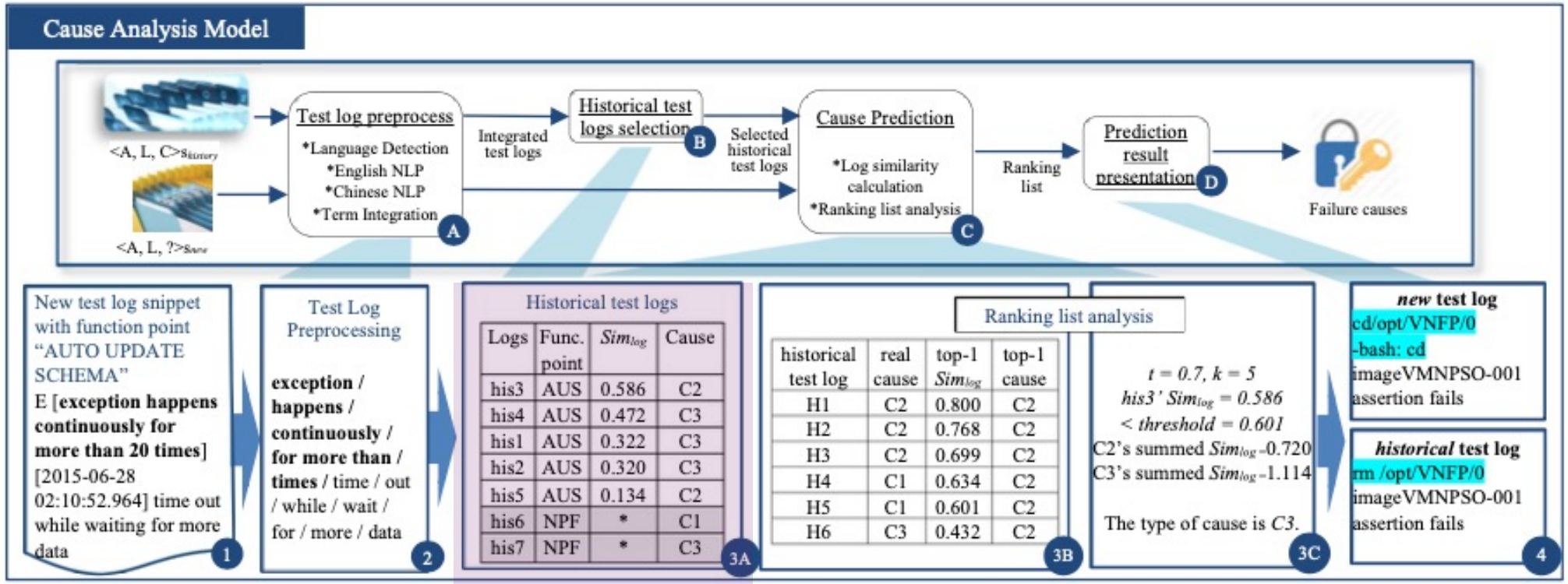


Task Formulation: Give <A, L, ?>, predict correct C

Evaluation: Test in two industrial testing projects at Huawei-Tech Inc.

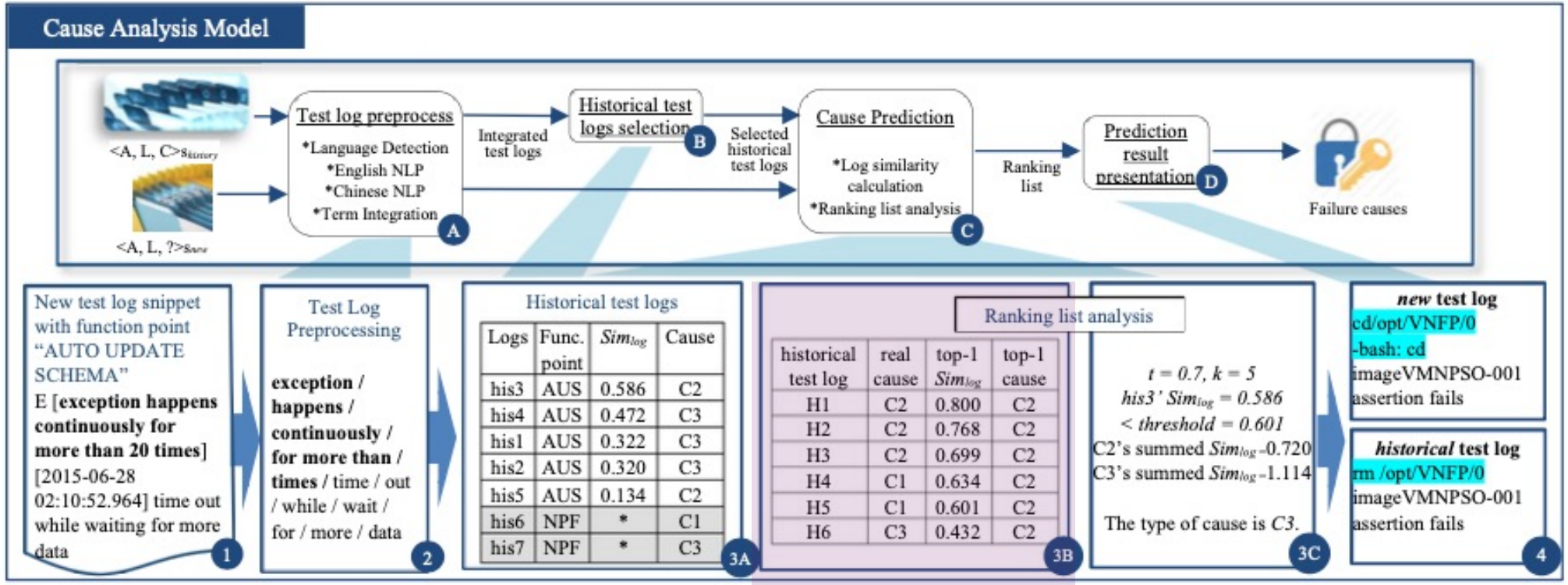
Jiang H, Li X, Yang Z, et al. What causes my test alarm? Automatic cause analysis for test alarms in system and integration testing[C]//2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). IEEE, 2017: 712-723.

Retrieval-based Methods



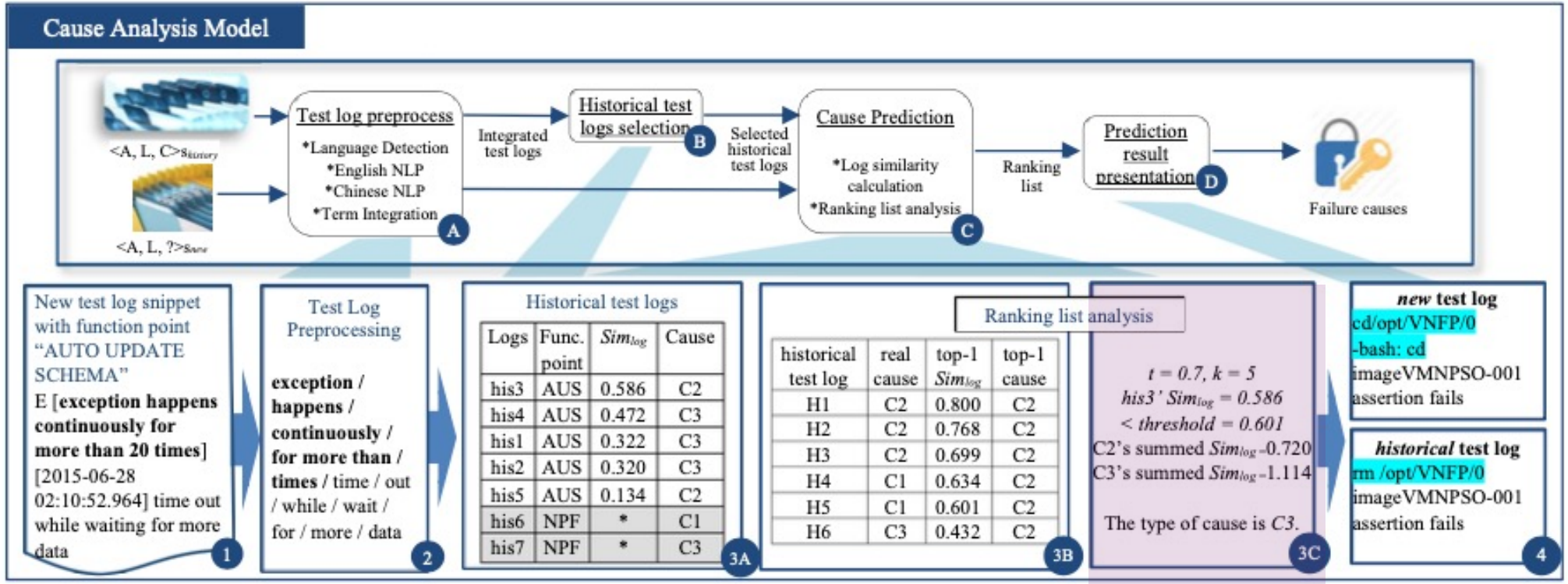
Selects historical test logs by examining the *function points of test scripts*, since the authors find that test scripts with the same function point usually target the same functionalities to check.

Retrieval-based Methods



Predict the cause of a new test alarm by first ranking the selected historical test logs according to their similarities with the new test log.

Retrieval-based Methods



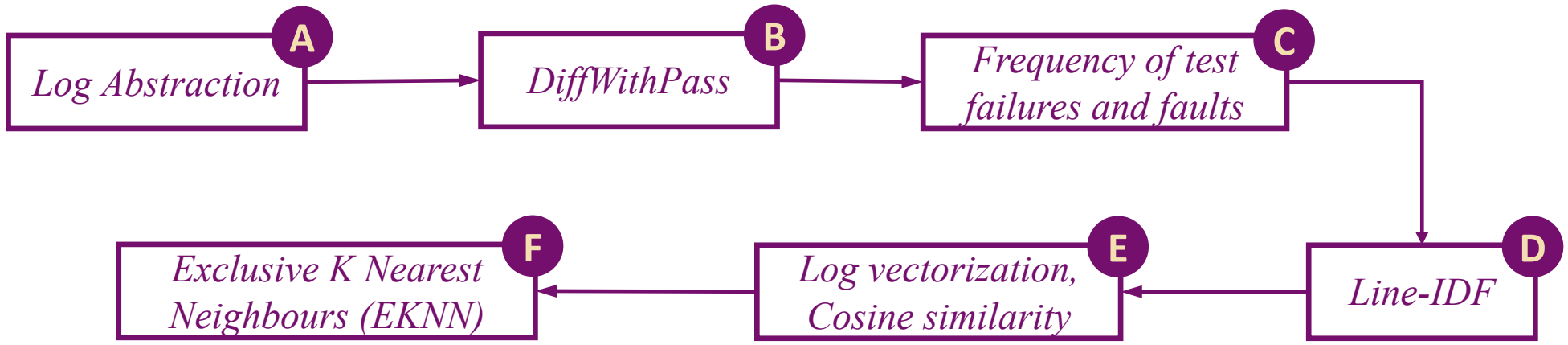
Normally predict the test alarm cause with the top-1 cause in the ranking list. If the similarity is low, use KNN strategy.

Retrieval-based Methods

Previous limitation and new challenge:

- The complex test environment introduces many non-product test failures. *#FaultsFound*
- The logs contain an overwhelming amount of detailed information. *#LogLinesFlagged*

Approach:



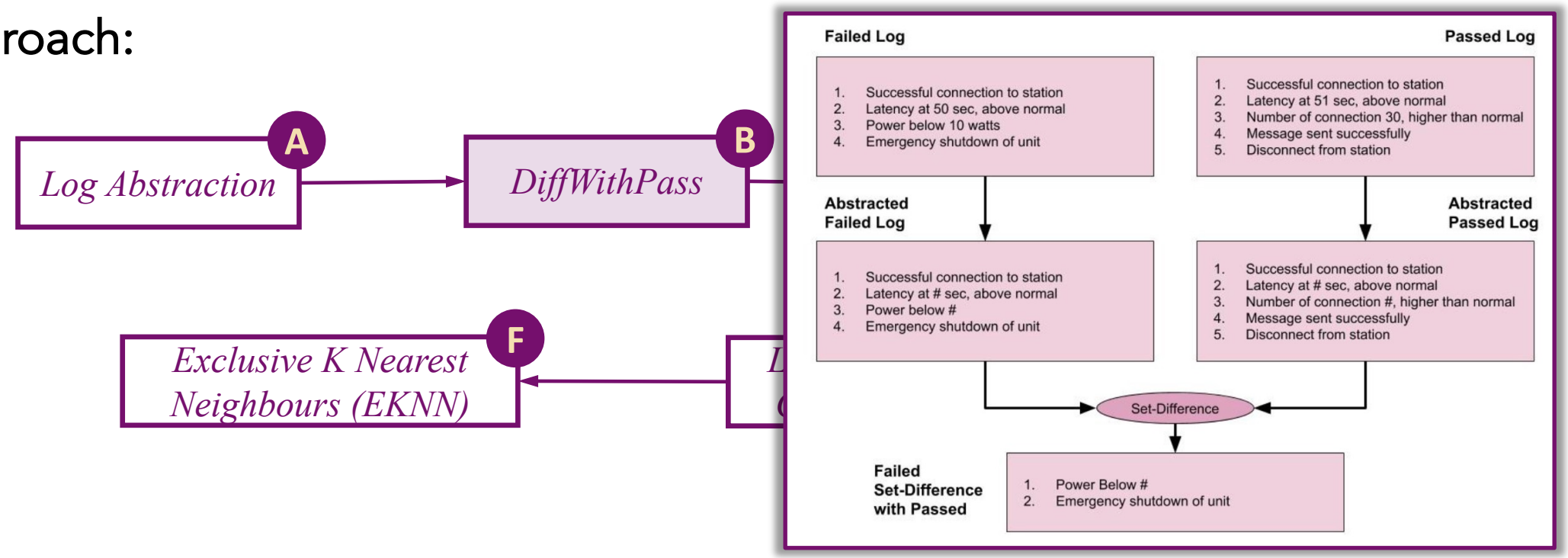
Amar A, Rigby P C. Mining historical test logs to predict bugs and localize faults in the test logs[C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 140-151.

Retrieval-based Methods

Previous limitation and new challenge:

- The complex test environment introduces many non-product test failures. *#FaultsFound*
- The logs contain an overwhelming amount of detailed information. *#LogLinesFlagged*

Approach:



Amar A, Rigby P C. Mining historical test logs to predict bugs and localize faults in the test logs[C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 140-151.

Retrieval-based Methods

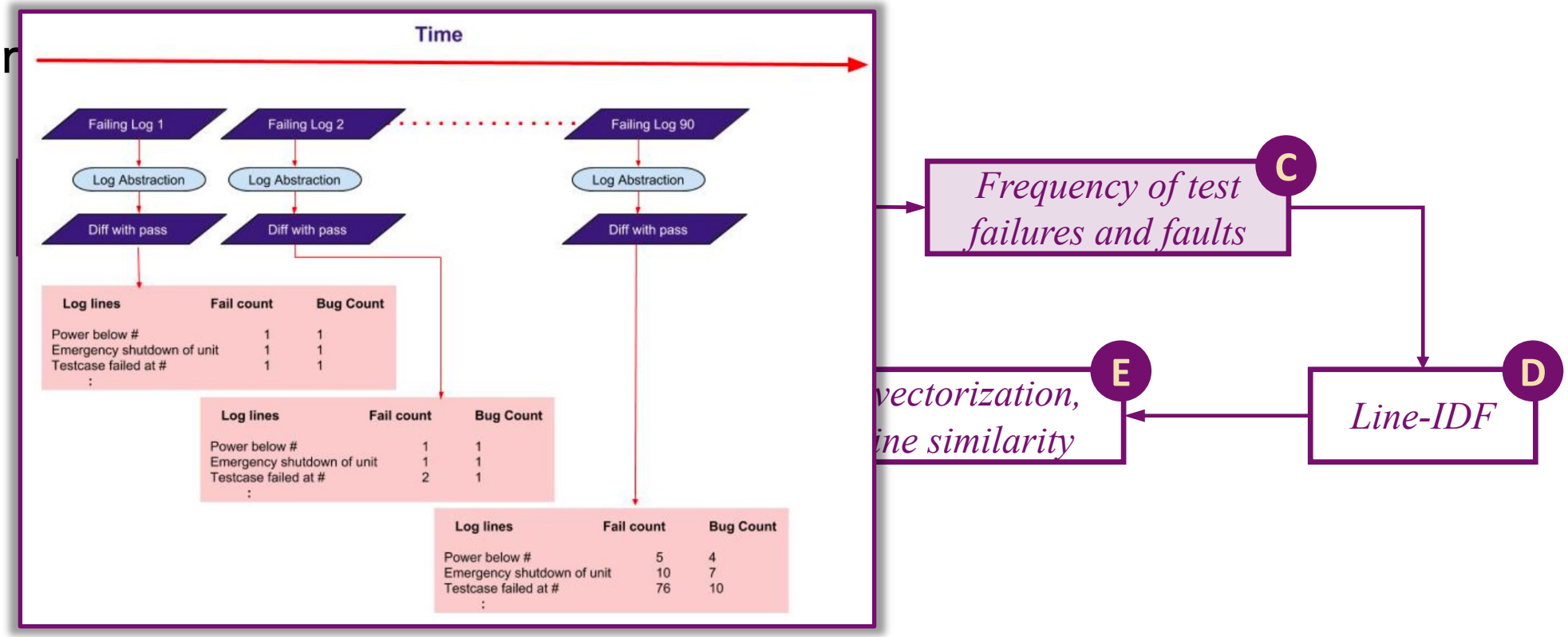
Previous limitation and new challenge:

- The complex test environment introduces many non-product test failures.
- The logs contain an overwhelming amount of detailed information.

#FaultsFound

#LogLinesFlagged

Appr



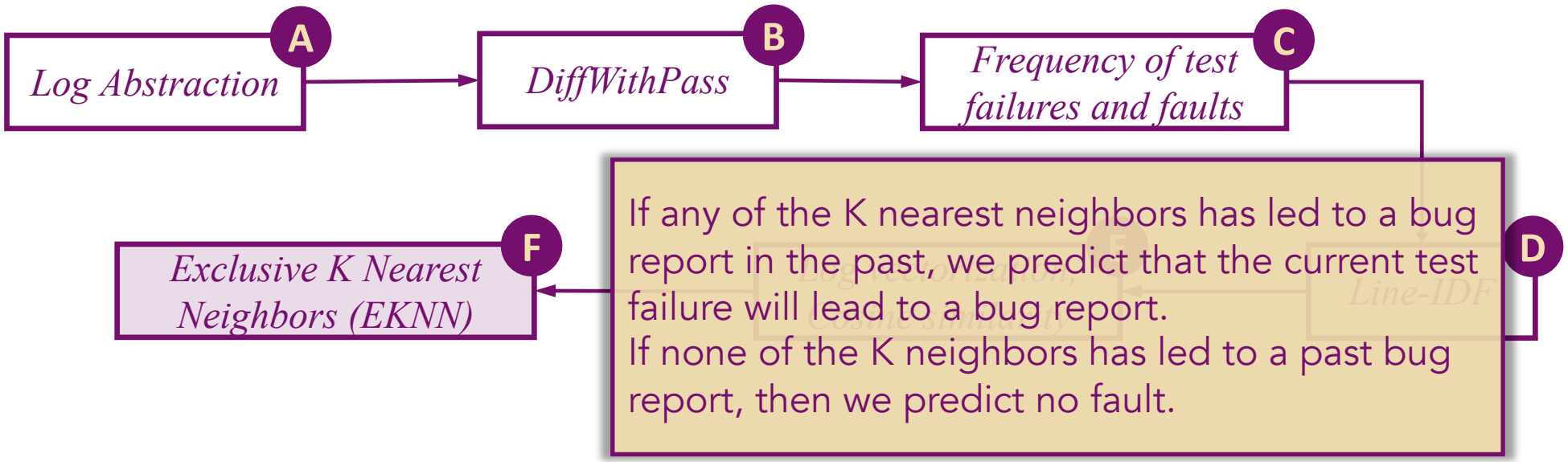
Amar A, Rigby P C. Mining historical test logs to predict bugs and localize faults in the test logs[C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 140-151.

Retrieval-based Methods

Previous limitation and new challenge:

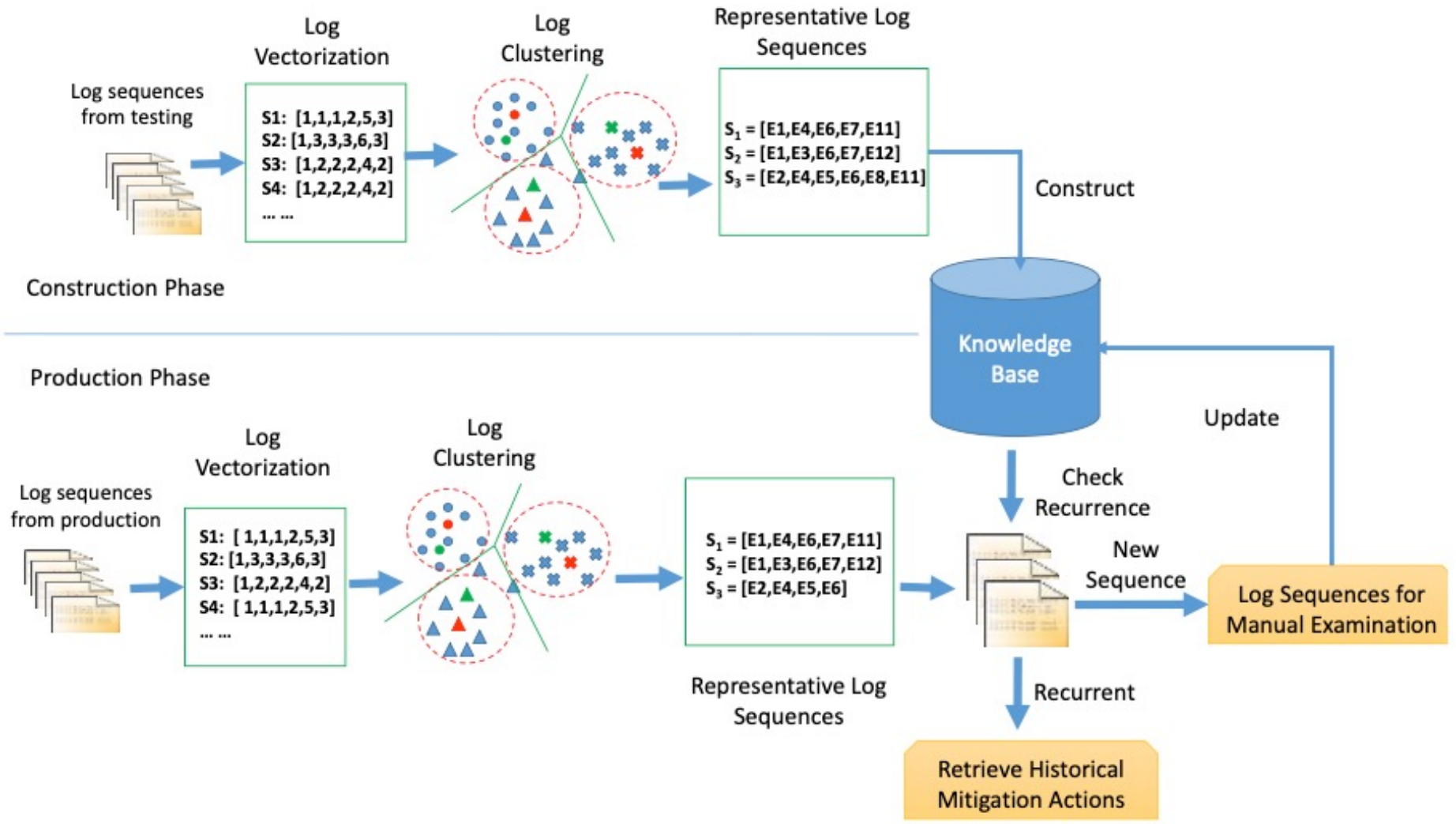
- The complex test environment introduces many non-product test failures. *#FaultsFound*
- The logs contain an overwhelming amount of detailed information. *#LogLinesFlagged*

Approach:



Amar A, Rigby P C. Mining historical test logs to predict bugs and localize faults in the test logs[C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 140-151.

Retrieval-based Methods



Lin Q, Zhang H, Lou J G, et al. Log clustering based problem identification for online service systems[C]//2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). IEEE, 2016: 102-111.

Conclusion

#1: Execution Replay Methods	• Infer execution flow • Trace back to software failure
#2: Model-based Methods	• Build reference model • Check violate
#3: Statistics-based Methods	• Statistical analysis
#4: Retrieval-based Methods	• Information Retrieval

Thank you!

香港中文大學
The Chinese University of Hong Kong