FSE 2024

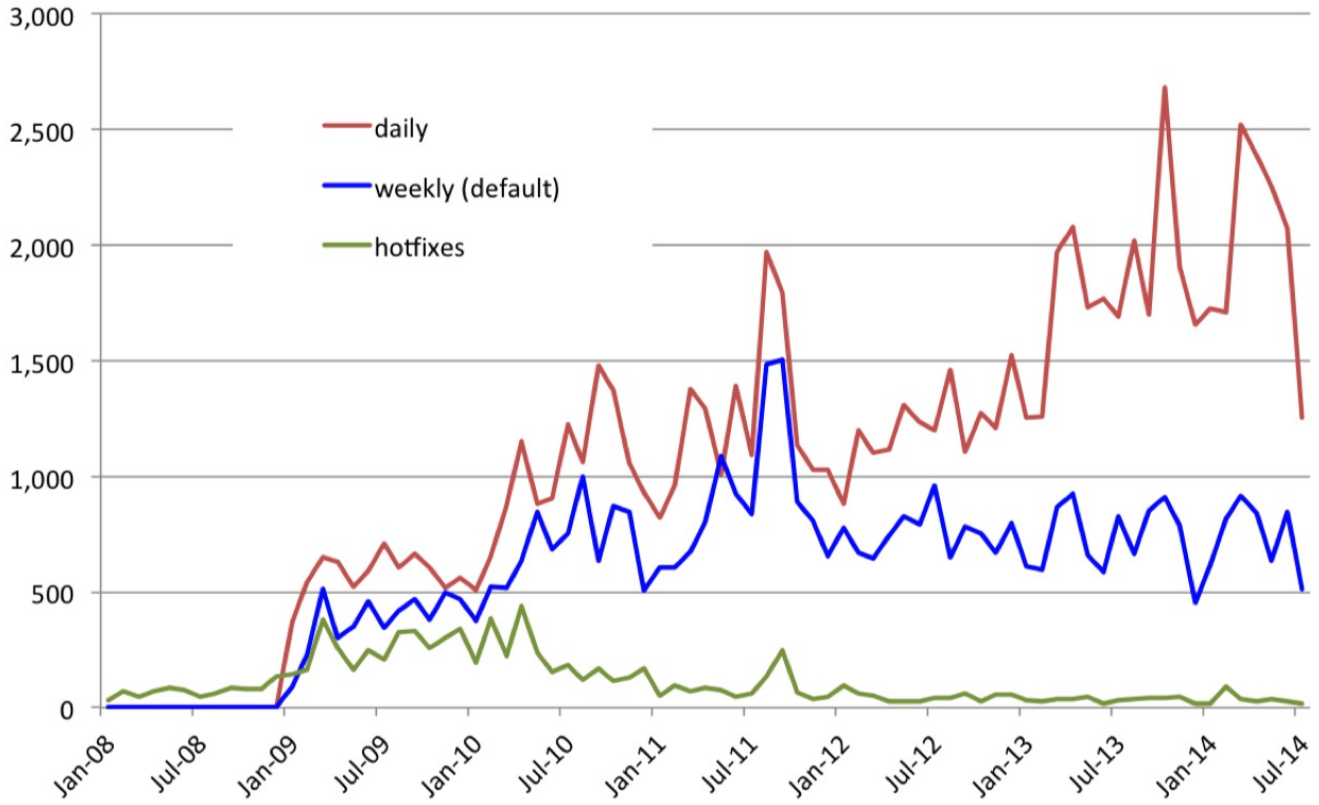
ChangeRCA: Finding Root Causes from Software Changes in Large Online Systems

Guangba Yu* Pengfei Chen* Zilong He* Qiuyu Yan†
Yu Luo† Fangyuan Li† Zibin Zheng*

*Sun Yat-sen University, China
† Tencent, China

Introduction

- Software change are **frequent and inevitable** in online systems
  - More than **2,000 deployment requests per day** in Facebook[1]

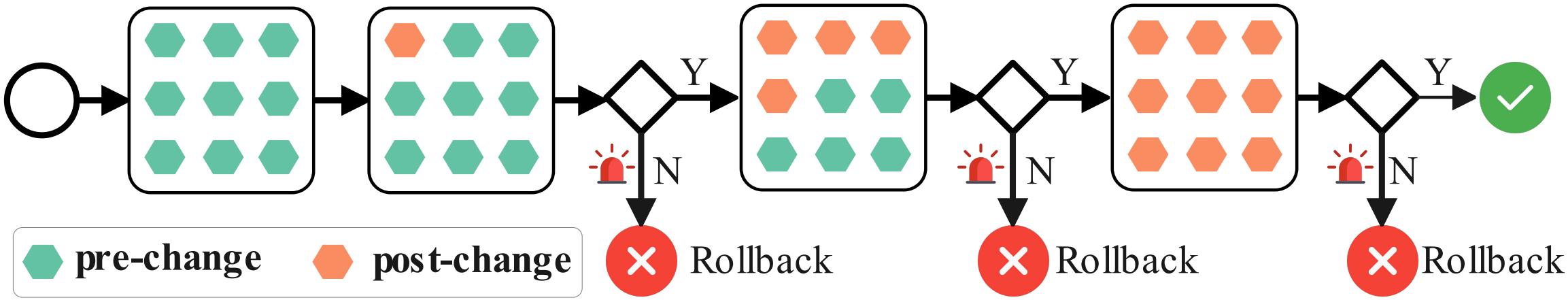


Line of Facebook deployment requests

[1] *Continuous Deployment at Facebook and OANDA [ICSE'16]*

Introduction

- **Canary change** is a widely-used change policy in online systems
  - Incrementally deploy new version to a selected group of users
  - Fast rollback if any defective changes are encountered



Workflow of a canary software change

Introduction

- Software change **failures are common** in online systems
  - More than **40% of incidents** are directly linked to software changes[1-2]

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*

Introduction

- Software change **failure is common** in online systems
  - More than **40% of incidents** are directly correlated to software changes[1-2]
  - Defective changes negatively **impact user experience and business revenue**
    - Facebook 'configuration change' caused users to be logged out unexpectedly[3]
    - Meta lost **$47.3 billion** in stock market value

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*
[3] *https://www.theverge.com/2021/1/23/22245842/facebook-logged-out-configuration-change-ios-app-security*

Introduction

- Software change **failure is common** in online systems
  - More than **40% of incidents** are directly correlated to software changes[1-2]
  - Defective changes negatively **impact user experience and business revenue**
    - Facebook 'configuration change' caused users to be logged out unexpectedly[3]
    - Meta lost **$47.3 billion** in stock market value

Identify defective changes promptly is important for minimizing negative impact

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*
[3] *https://www.theverge.com/2021/1/23/22245842/facebook-logged-out-configuration-change-ios-app-security*

Introduction

- Many previous work on identifying bugs before deployment
  - Software test [8, 21, 45, 58, 59]
  - Reliability auditing [4, 49, 57]
  - System verification [5, 32, 62]
- **Defects still emerge in production environment** due to the difference between testing and production environments
  - Service scale
  - Hardware heterogeneity

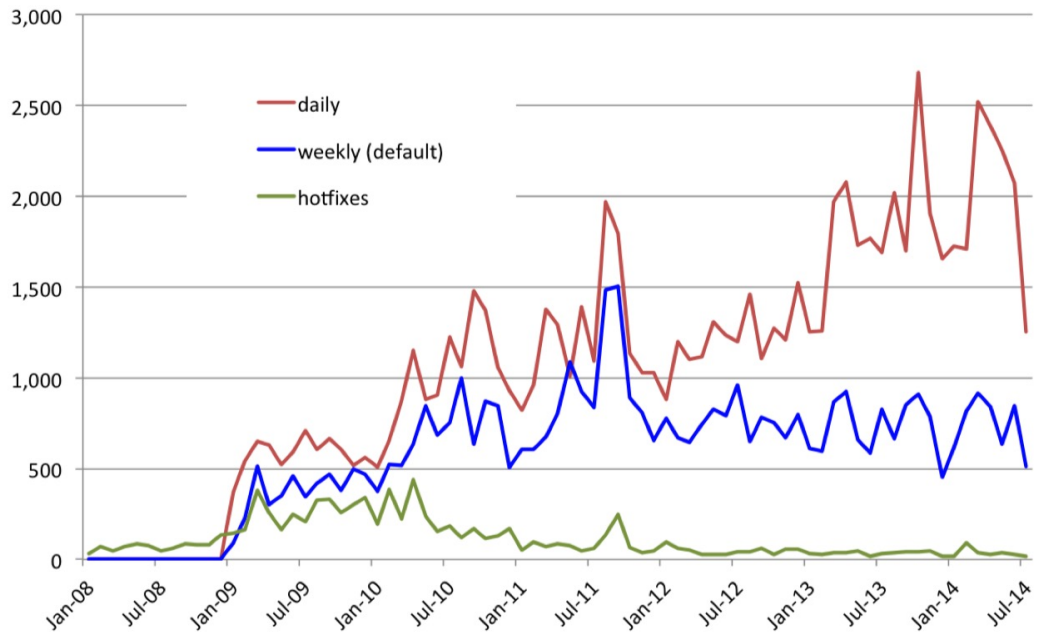
Introduction

- Many previous work on identifying bugs before deployment
  - Software test [8, 21, 45, 58, 59]
  - Reliability auditing [4, 49, 57]
  - System verification [5, 32, 62]
- **Defects still emerge in production environment** due to the difference between testing and production environments

We need to identify defective changes after deployment

Challenges

- Identify defective change in large-scale systems is **challenging**
  1. **Large-scale search space** of suspicious changes
     - Multiple changes co-exist in a system

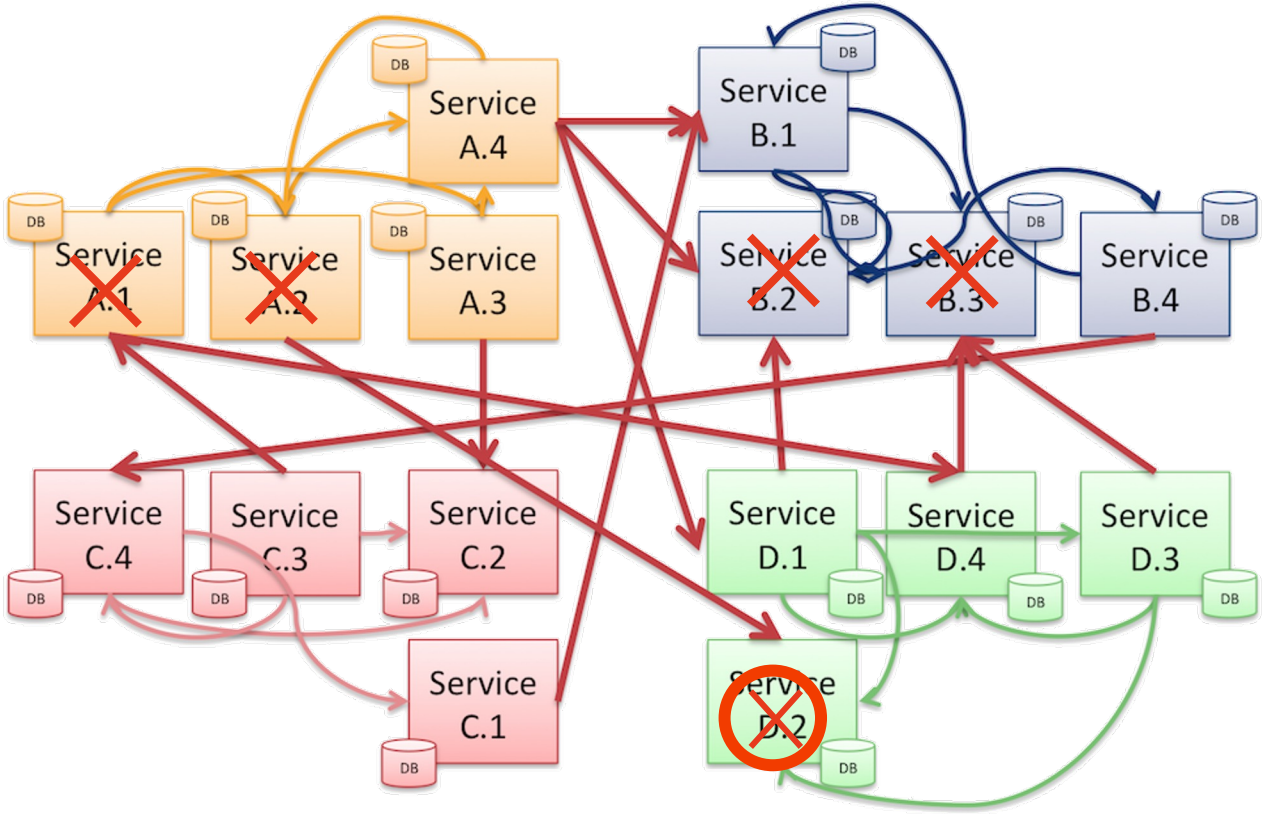


More than 2,000 deployment requests per day in Facebook[1]

[1] Continuous Deployment at Facebook and OANDA [ICSE'16]

Challenges

- Identify defective change in large-scale systems is **challenging**
  1. Large-scale search space of suspicious changes
  2. **Anomalous propagation** of defective changes leads to many abnormal changes
     - Overcome anomalous propagation to **exclude false positive change**

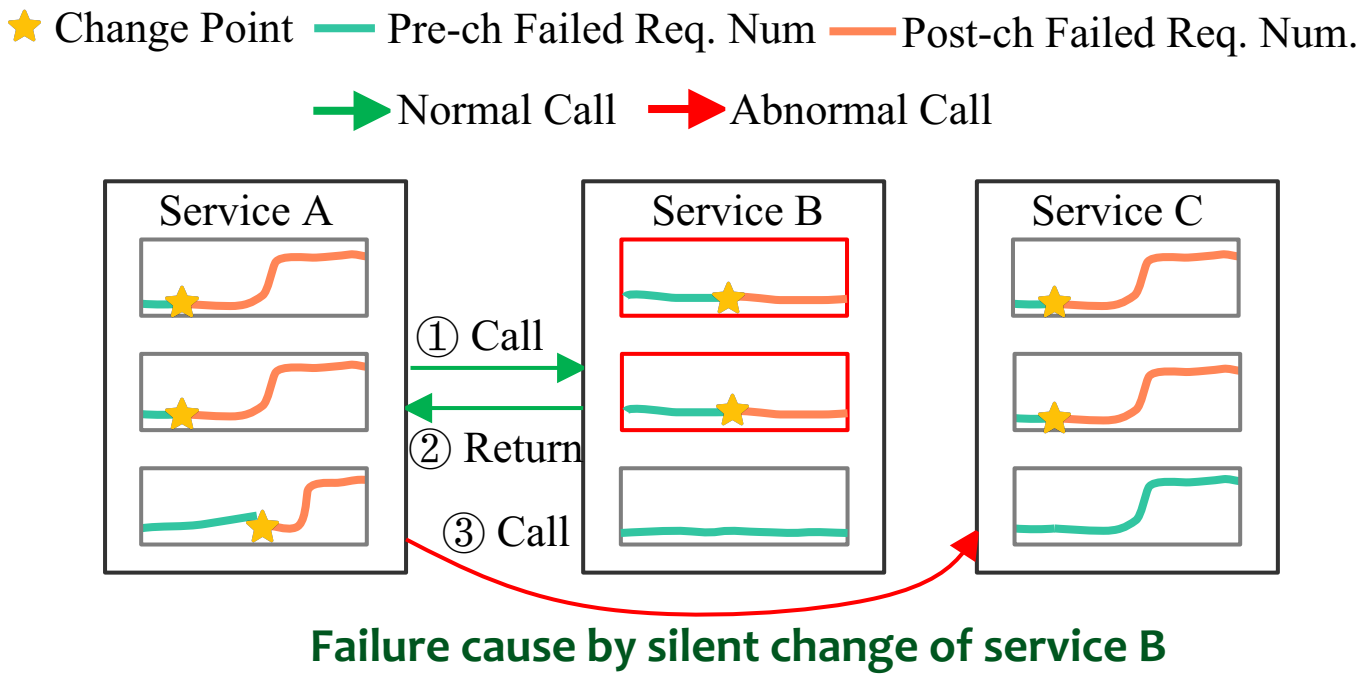


Abnormal change caused by propagation

Real defective change

Challenges

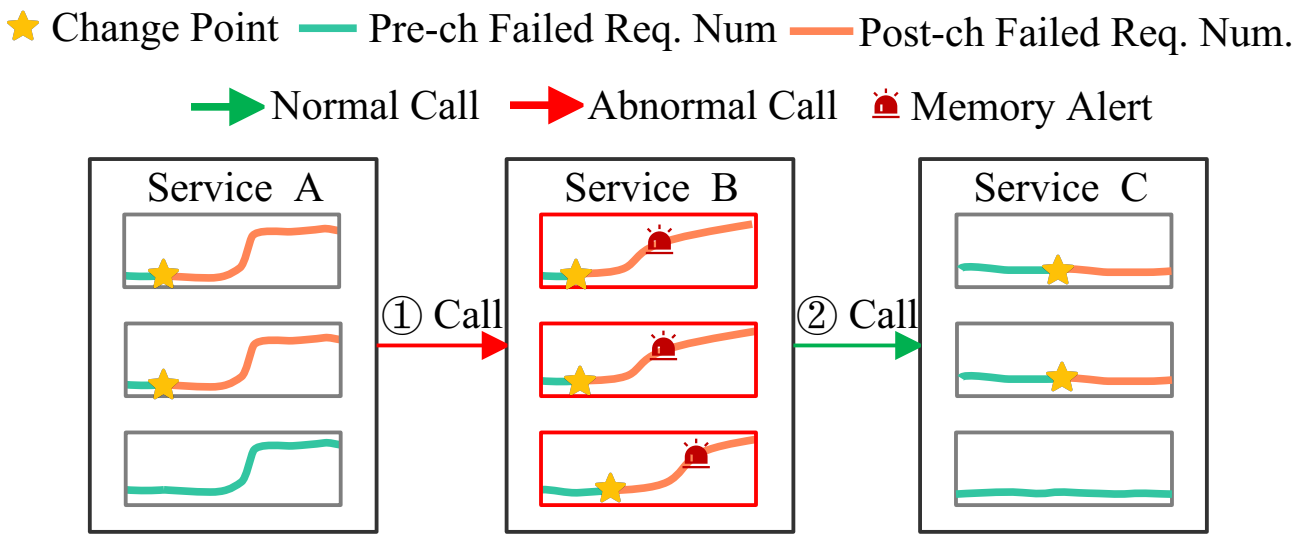
- Identify defective change in large-scale systems is **challenging**
  1. Large-scale search space of suspicious changes
  2. Anomalous propagation of defective changes
  3. **Silent defective changes**' KPIs are performing normally
     - Difficult to identify them by considering **only KPIs**



Failure cause by silent change of service B

Challenges

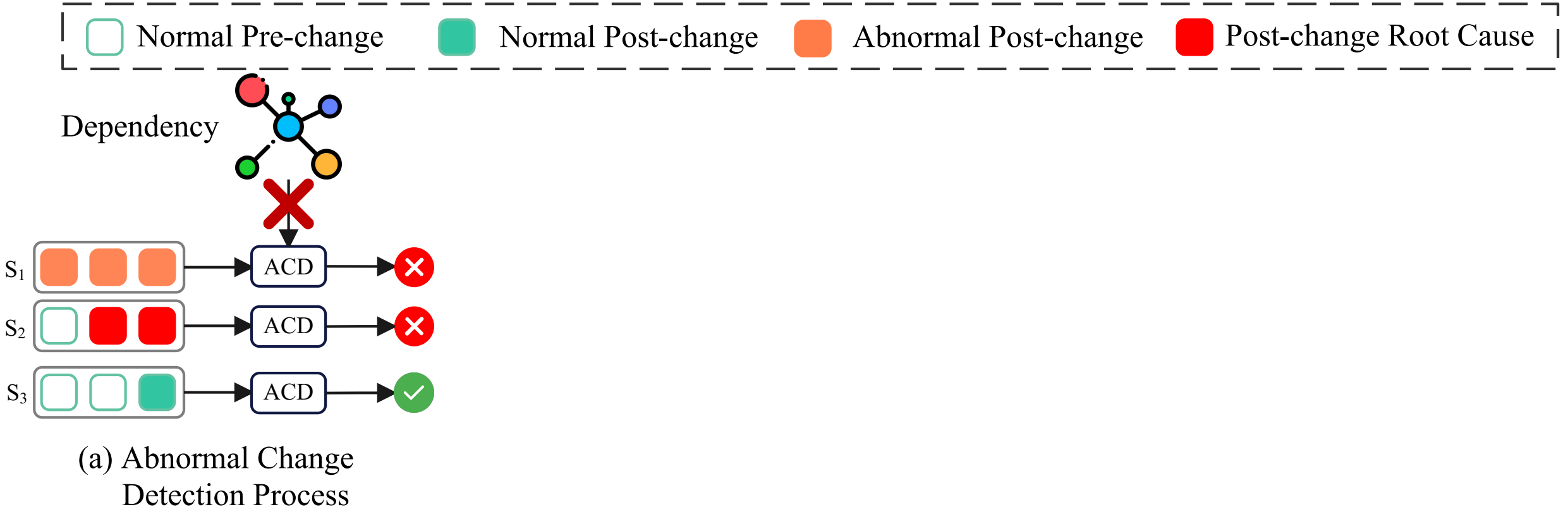
- Identify defective change in large-scale systems is **challenging**
  1. Large-scale search space of suspicious changes
  2. Anomalous propagation of defective changes
  3. Silent defective changes' KPIs are performing normally
  4. Some changes emerge abnormal behavior **after the change completion**
     - e.g., memory leak



Memory leak failure cause by backend change of service B

Existent Approach

- Abnormal change detection (ACD): individually check changes **within a single service** [1-2]

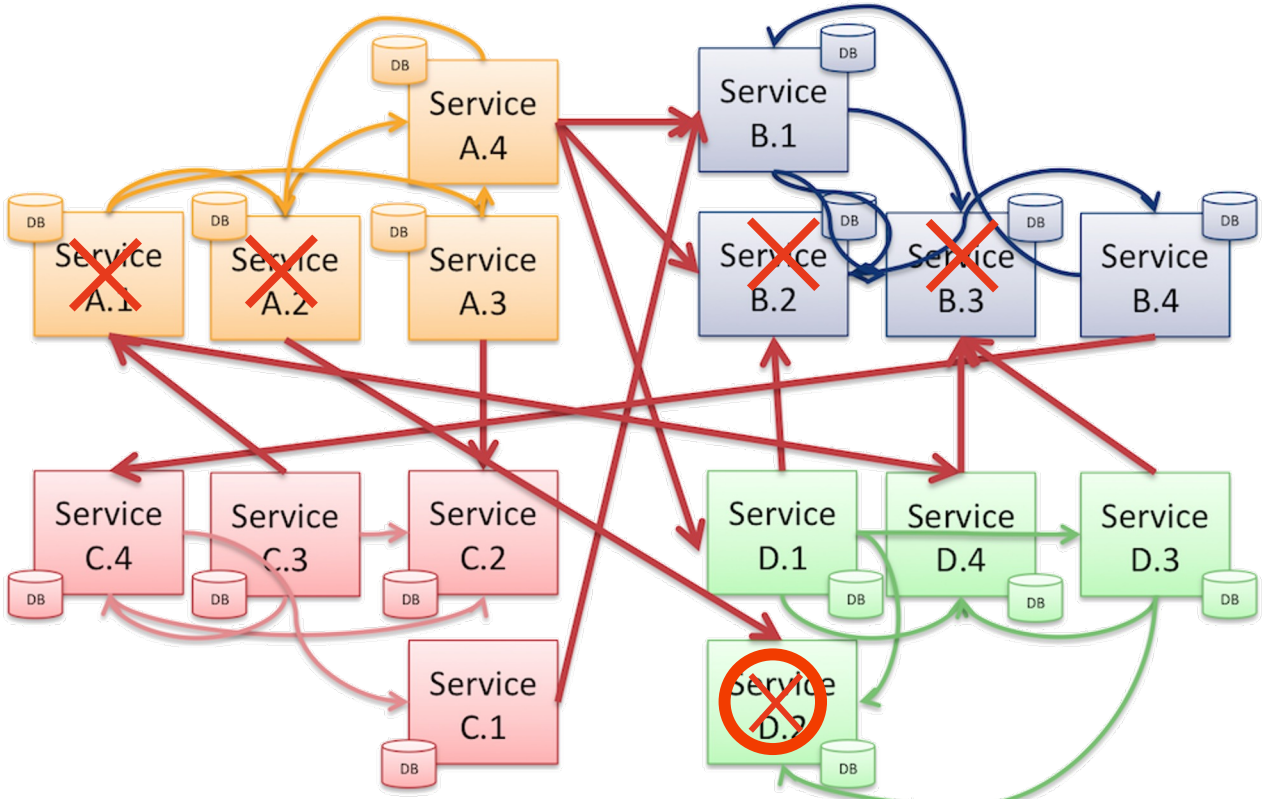


(a) Abnormal Change Detection Process

[1] *Rapid and robust impact assessment of software changes in large internet-based services [CoNEXT'15]*
[2] *Identifying bad software changes via multimodal anomaly detection for online service systems [FSE'21]*

Existent Approach

- Abnormal change detection (ACD): assess individual changes **within a single service**
  - **Overlook abnormal propagation** of defective changes
    - Disregard service dependencies
    - Lead to multiple **false alarms**

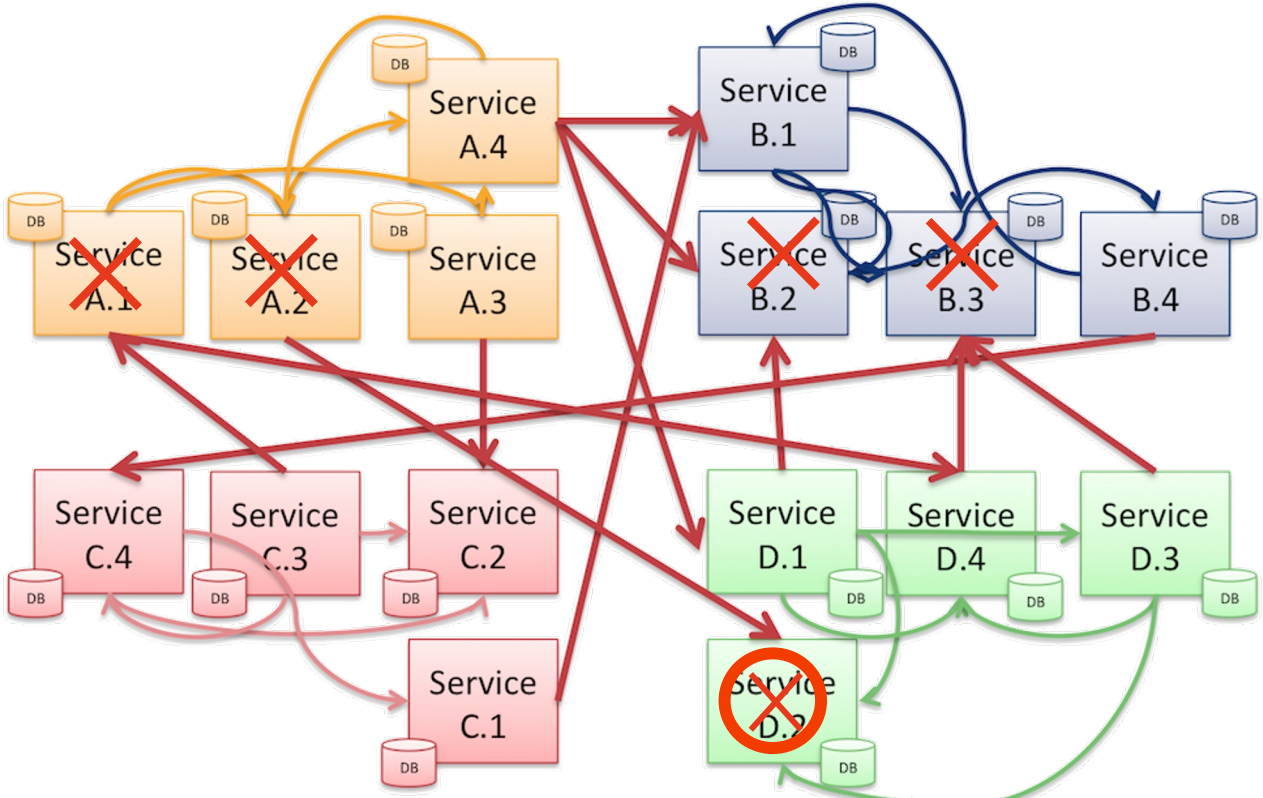


Abnormal change caused by propagation

Real defective change

Existent Approach

- Abnormal change detection (ACD): assess individual changes **within a single service**
  - **Overlook anomalous propagation** of defective changes
    - Disregard service dependencies
    - Lead to multiple **false alarms**



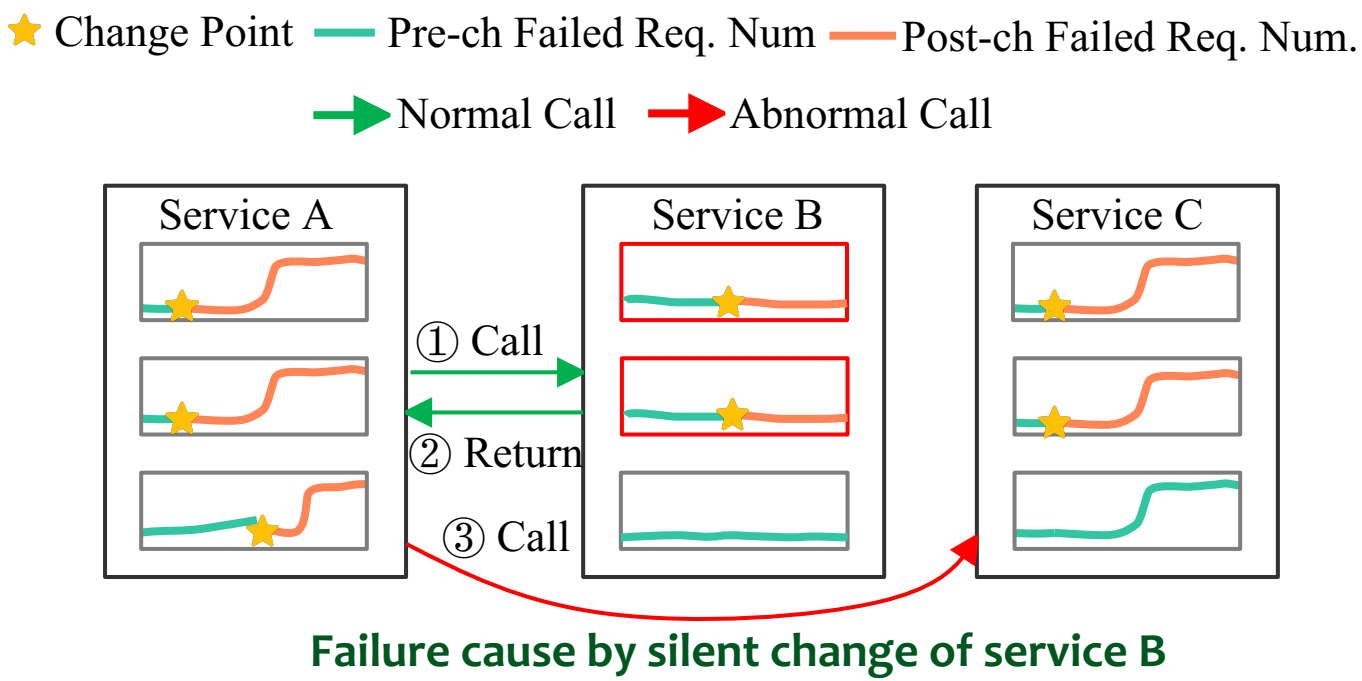
🤯 ***Unable to handle challenge 1 and 2***

✕ Abnormal change caused by propagation

⊗ Real defective change

Existent Approach

- Abnormal change detection (ACD): assess individual changes **within a single service**
  - Overlook anomalous propagation of defective changes
  - **Unable to handle silent** defective changes
    - Effectiveness is dependent on abnormal KPI performance

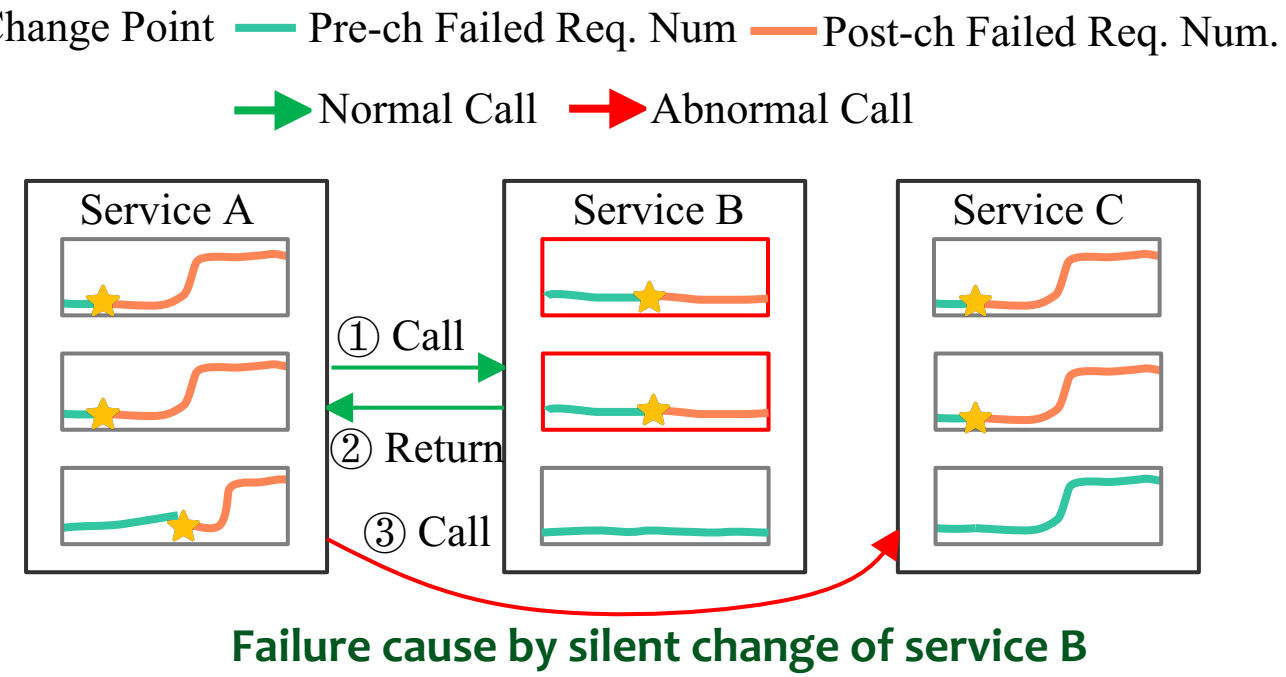


Failure cause by silent change of service B

Existent Approach

- Abnormal change detection (ACD): assess individual changes **within a single service**
  - Overlook anomalous propagation of defective changes
  - Unable to handle silent defective changes
    - Dependent on abnormal KPI performance
    - Lead to multiple false negative

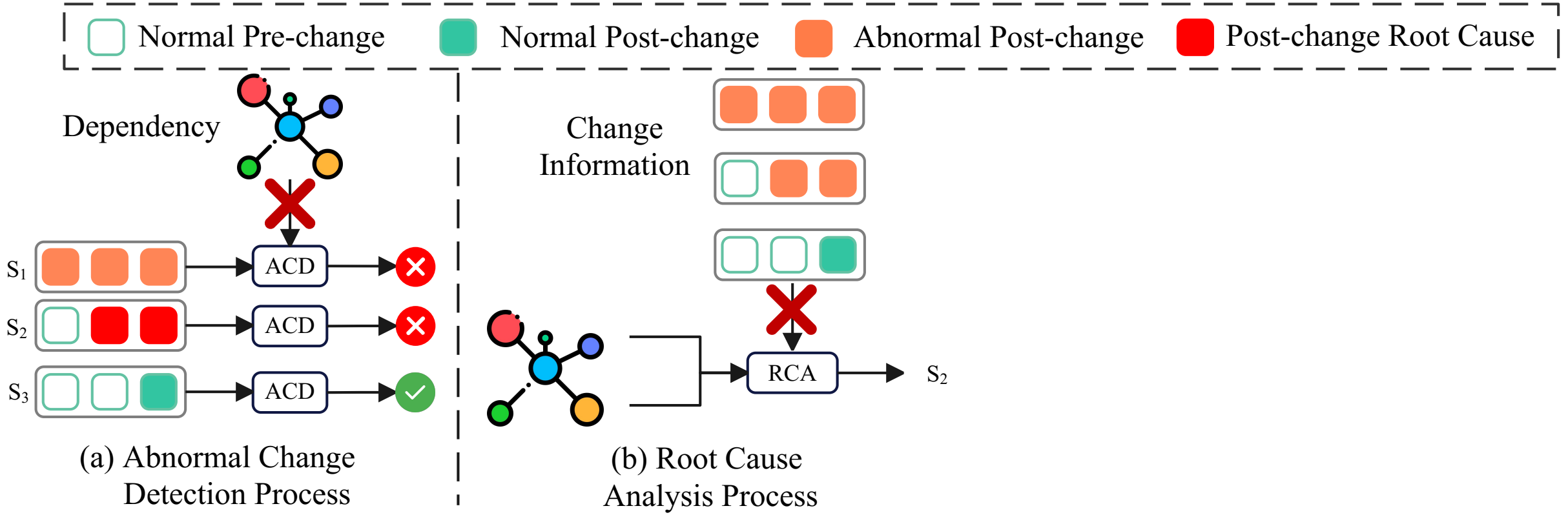
Unable to handle challenge 3 and 4



Failure cause by silent change of service B

Existent Approach

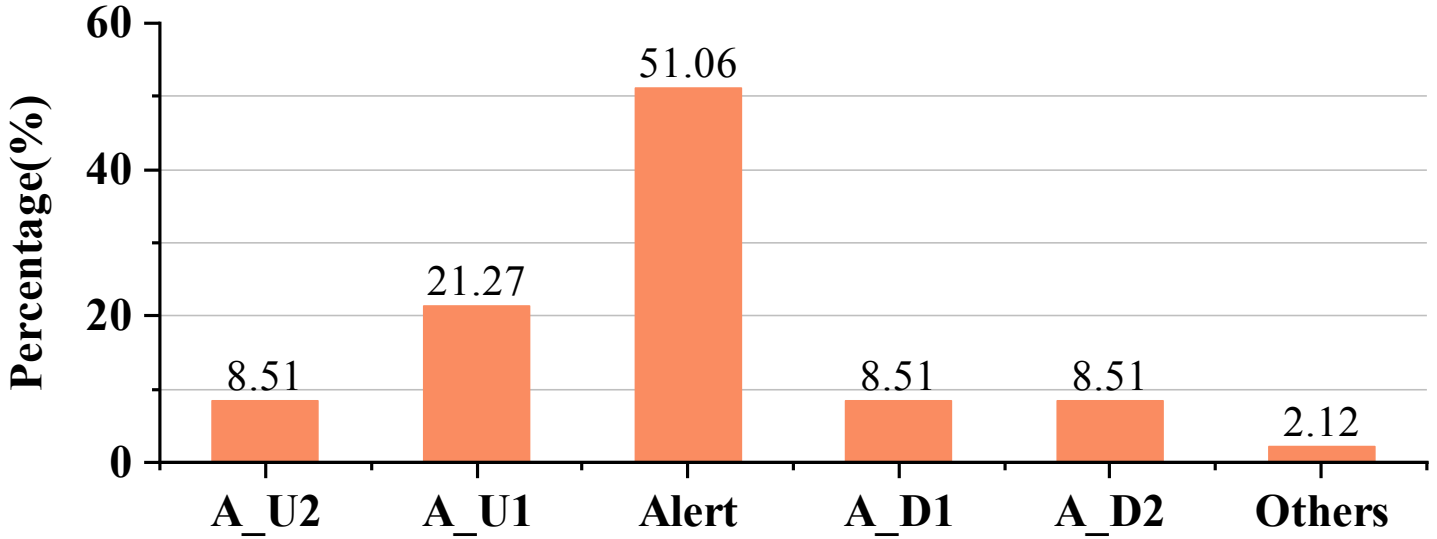
➢ **Root cause analysis (RCA):** localize root causes **at service or service instance level** [1-2]



[1] *Graph based Incident Extraction and Diagnosis in Large-Scale Online Systems [ASE'22]*
[2] *MicroRank: End-to-End Latency Issue Localization with Extended Spectrum Analysis in Microservice Environments [WWW'21]*

Existent Approach

- **Root cause analysis (RCA):** localize root causes **at service or service instance level**
  - State-of-the-art RCA[1] effectively **narrows down the search scope** for defective changes



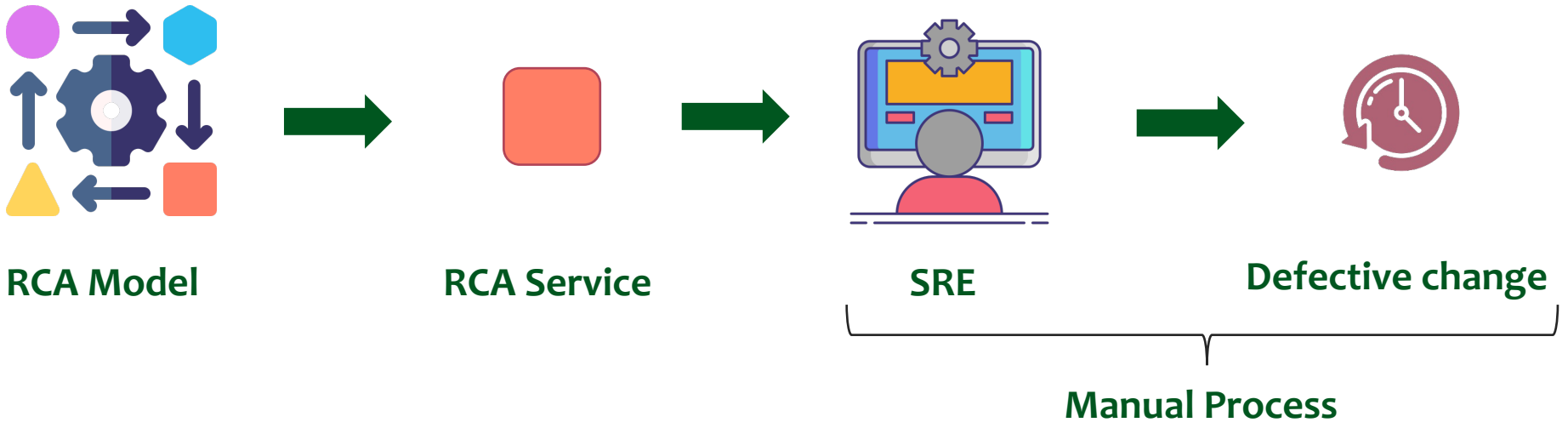
Differences between RCA result and defective change service on change failure dataset from Wechat

97% of defective change services are within 2 depth ranges of RCA results

[1] *Graph based Incident Extraction and Diagnosis in Large-Scale Online Systems [ASE'22]*
2 *Study on a change failure dataset from Wechat*

Existent Approach

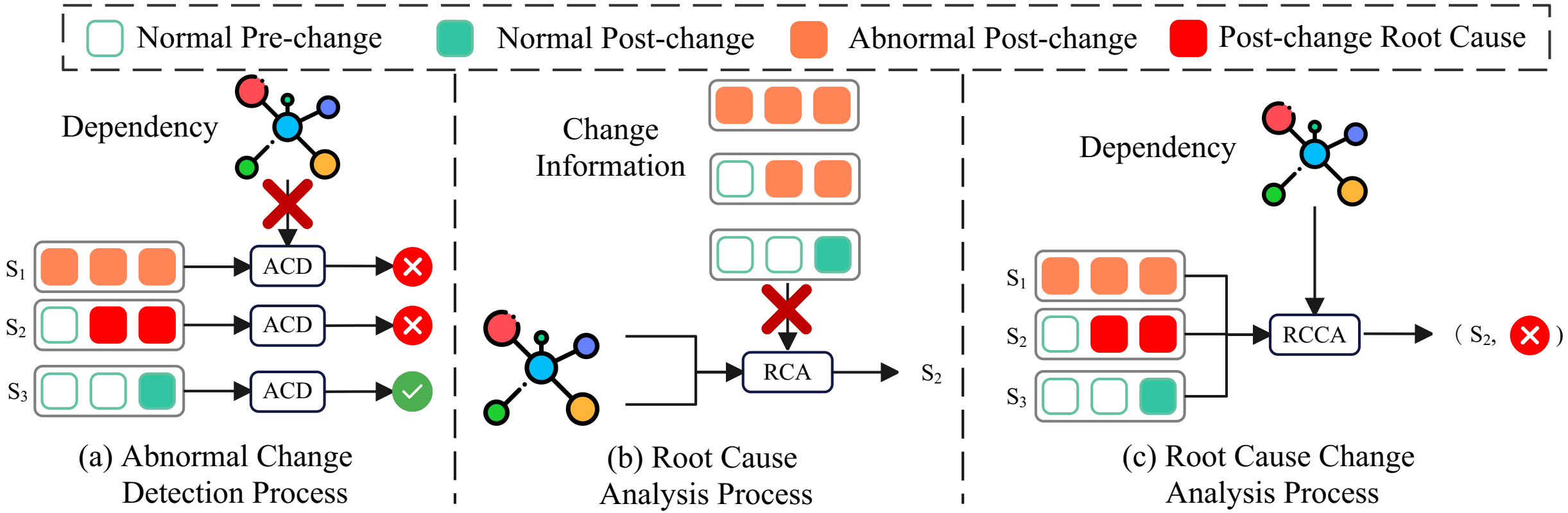
- **Root cause analysis (RCA):** localize root causes at service or service instance level
  - State-of-the-art RCA effectively narrow down the search scope for defective changes
  - **Overlook corresponding change data** (e.g., change flow)



SREs need to localize defective change manually

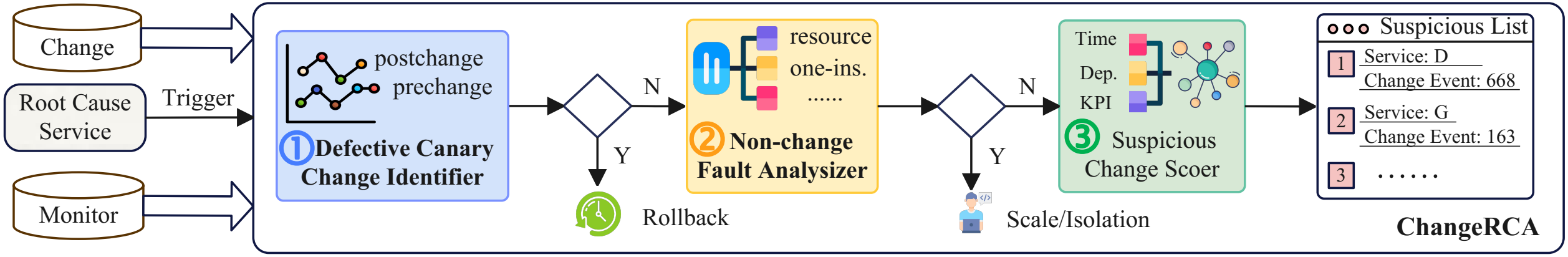
RCCA Approach

- **Root cause change analysis (RCCA):** identify defective changes responsible for failures
  - Consider both **service dependency and change process information**



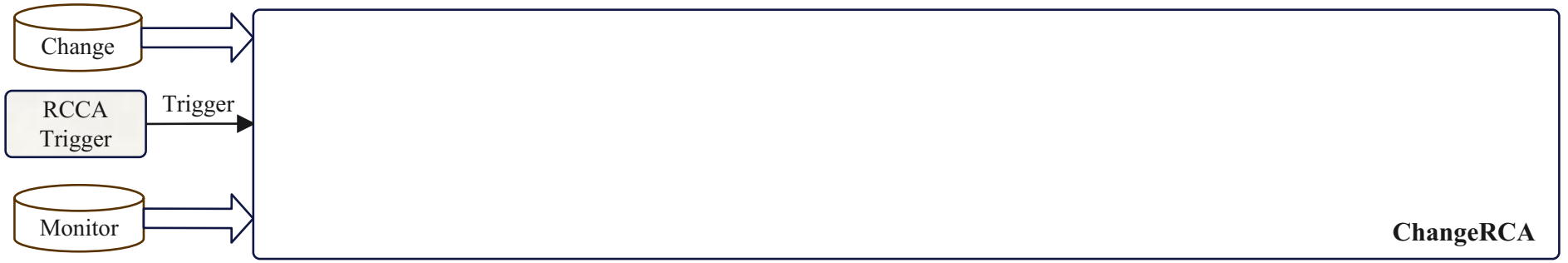
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - Make RCCA concept in practice
  - **Input**: RCA service, KPI data, change processes, dependency graph
  - **Output**: defective changes



ChangeRCA Framework

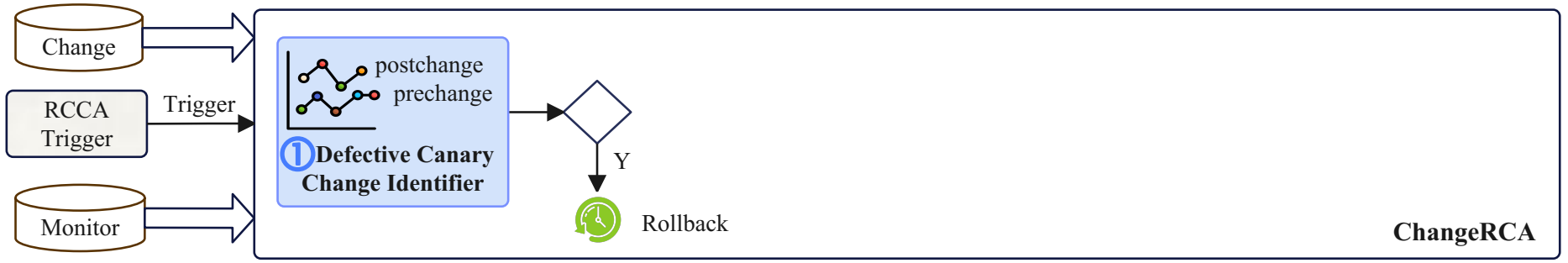
- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - Trigger by existent RCA approach, take root cause $\boldsymbol{S}_{rca}$ as input
  - Default RCA approach is GIED[1]
    - Adopts a GNN model and a PageRank algorithm to localize suspicious services
    - GIED has been proven effective in industrial systems



[1] *Graph based Incident Extraction and Diagnosis in Large-Scale Online Systems [ASE'22]*

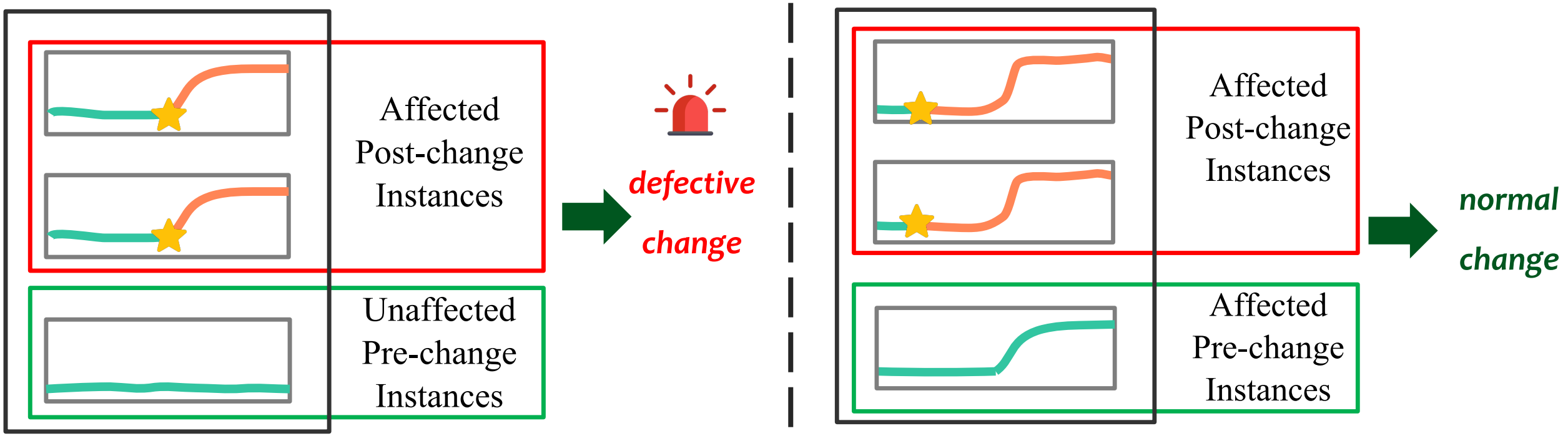
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify **defective canary change** of S_{rca} based DiD → ***Canary change***



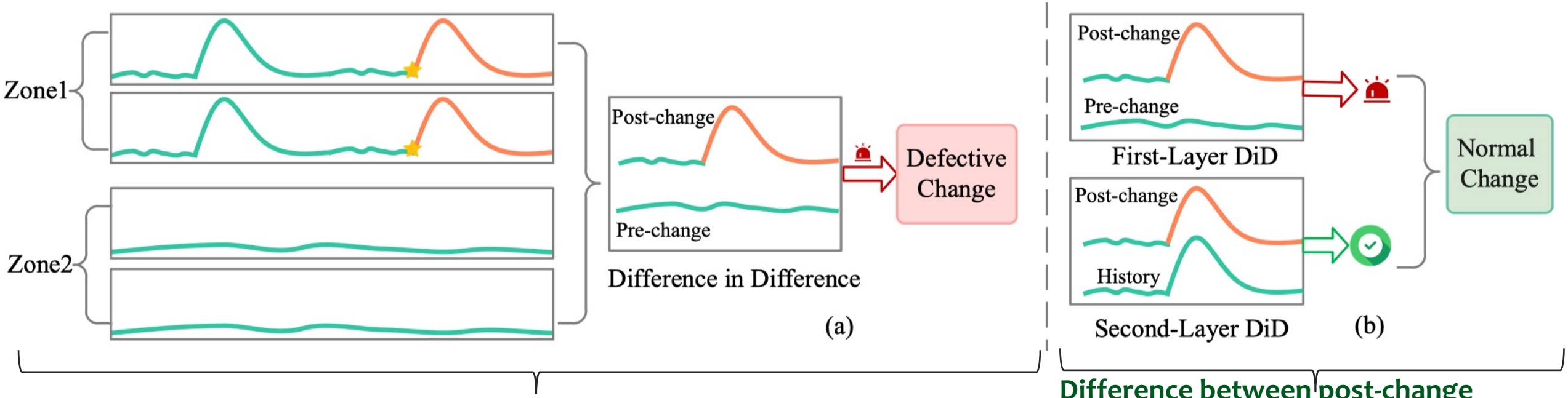
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify **defective canary change** of S_{rca} based DiD → **Canary change**
    - **Core idea**: If an incident is caused by defective change, it only **affects post-change instances** while leaving **pre-change instances unaffected**



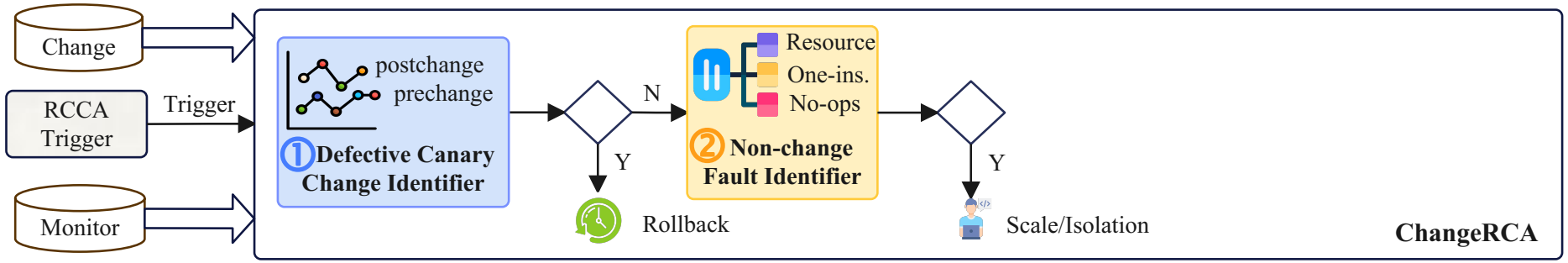
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify **defective canary change** of S_{rca} based DiD → ***Canary change***
    - **Core idea**: If an incident is caused by defective change, it only **affects post-change instances** while leaving **pre-change instances unaffected**
    - **Cascading DiD framework** consists of two layers of DiD



ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify defective canary change of S_{rca} based DiD → ✕
  - ② **Exclude non-change fault** based on pre-defined rules
    - Resource fault → ***Scale Up***
    - One-instance Fault → ***Isolation***
    - Non-ops Fault → ***Non-ops***

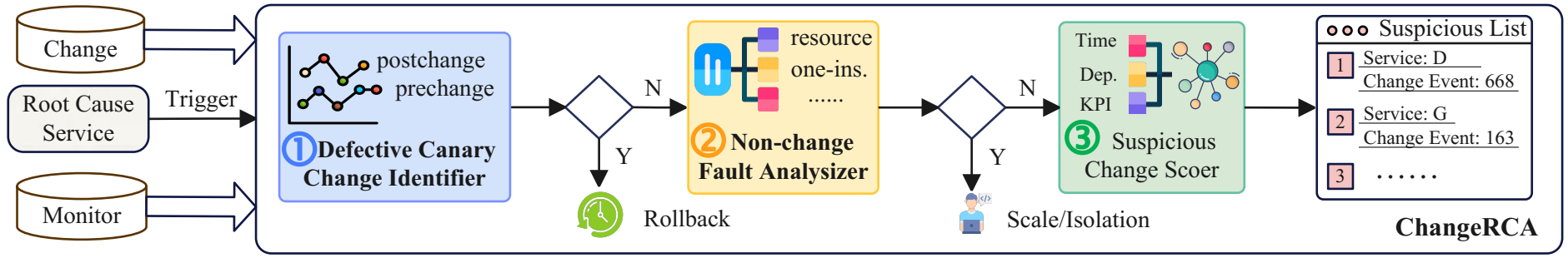


ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify defective canary change of S_{rca} based DiD
  - ② Exclude non-change fault based on pre-defined rule
  - ③ **Localize suspicious changes** within dependency graph of S_{rca}

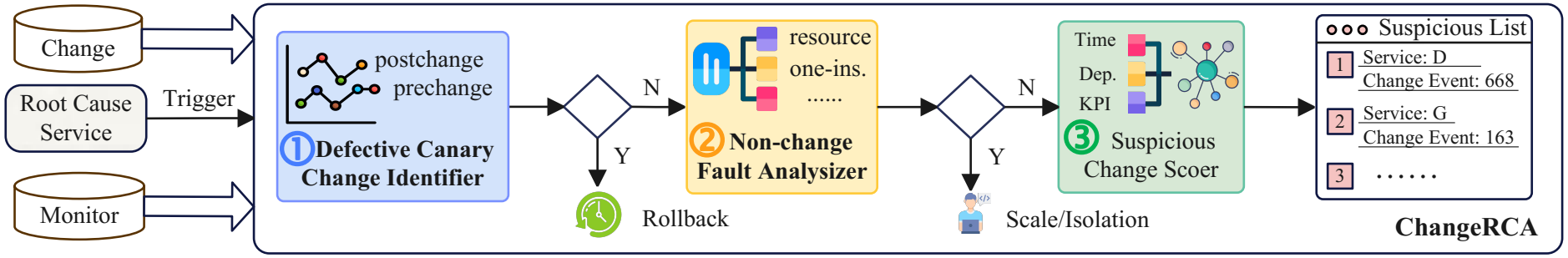
SRE Viewpoint

Prioritize investigating whether an incident is attributed to a defective change



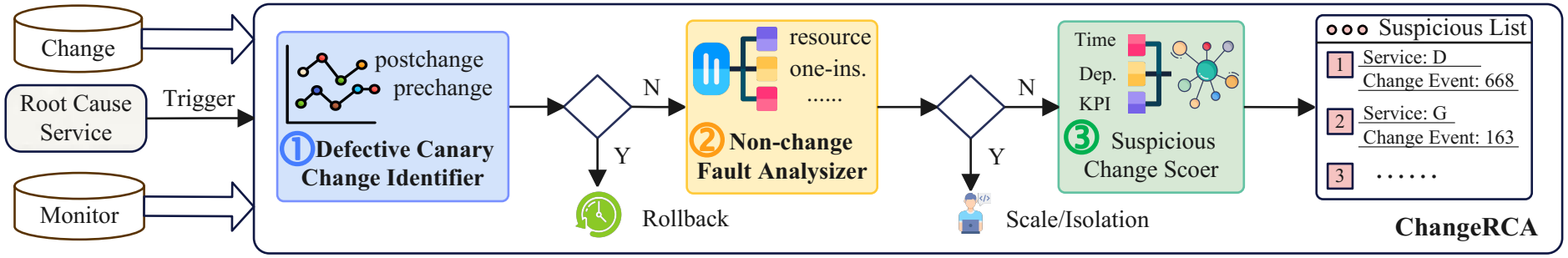
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify defective canary change of S_{rca} based DiD
  - ② Exclude non-change fault based on pre-defined rule
  - ③ **Localize suspicious changes** within dependency graph of S_{rca}
    - KPI Scorer: Services with **more abnormal post- instances and less pre- instances** are more suspicious



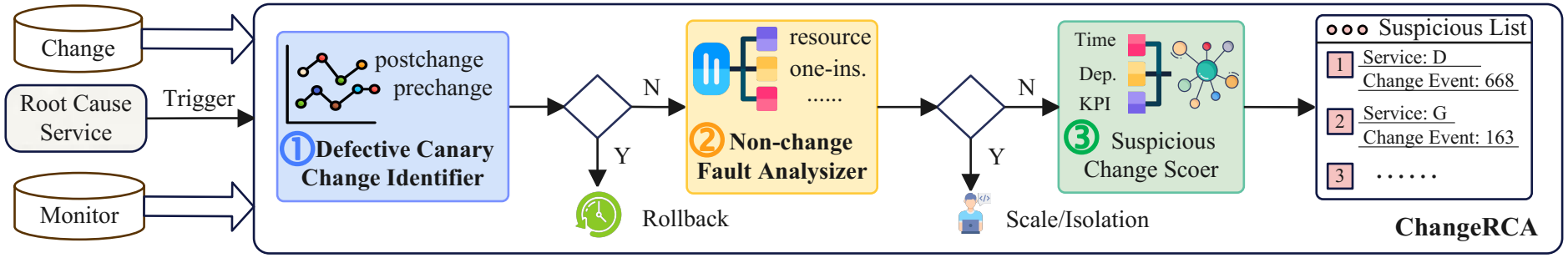
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify defective canary change of S_{rca} based DiD
  - ② Exclude non-change fault based on pre-defined rule
  - ③ **Localize suspicious changes** within dependency graph of S_{rca}
    - KPI Scorer: Services with **more abnormal post- instances and less pre- instances** are more suspicious
    - Dependency Scorer: Services **closer to S_{rca}** are more suspicious



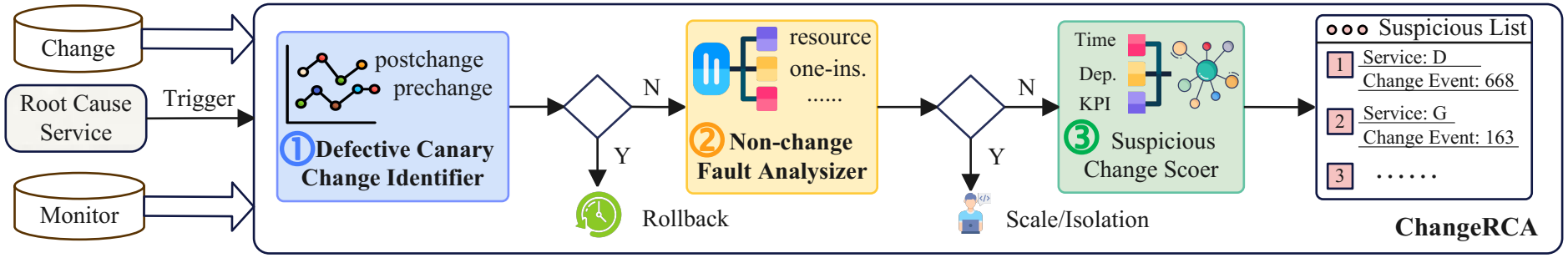
ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify defective canary change of S_{rca} based DiD
  - ② Exclude non-change fault based on pre-defined rule
  - ③ **Localize suspicious changes** within dependency graph of S_{rca}
    - KPI Scorer: Services with **more abnormal post- instances and less pre- instances** are more suspicious
    - Dependency Scorer: Services **closer to S_{rca}** are more suspicious
    - Time Scorer: Changes **closer to failure time** are more suspicious



ChangeRCA Framework

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - ① Identify defective canary change of S_{rca} based DiD → ✕
  - ② Exclude non-change fault based on pre-defined rule → ✕
  - ③ **Localize suspicious changes** within dependency graph of S_{rca}
    - KPI Scorer: Services with **more abnormal post- instances and less pre- instances** are more suspicious
    - Dependency Scorer: Services **closer to S_{rca}** are more suspicious
    - Time Scorer: Changes **closer to failure time** are more suspicious
    - Employ a hierarchical weighted average approach to combine scores → ***Suspicious list***



Experiment

Change Failure Dataset

- **Real-world change failure** dataset from **WeChat** (30 Cases)
  - Backend change
  - Resource change
  - Configuration change
  - Model change
- **Microserivce benchmark** dataset (51 Cases) [1]

Change Type	Fault Type
Defective Backend Change	Miss Func. Call, Wrong Except. Handle, Miss Param. Value, Wrong Return Value, Wrong Sql, Wrong Cache, Wrong Param. Order
Defective Config. Change	Wrong Port, Config. File Inconsistent, Wrong Access Key
Defective Resource Change	Memory Leak, Unsatisfactory Resource Allocation

[1] *How Bad Can a Bug Get? An Empirical Analysis of Software Failures in the OpenStack Cloud Computing Platform [FSE'19]*

Experiment

- Research question
  - RQ1: How **effective** is ChangeRCA in RCCA?
  - RQ2: How useful are **different scorers** for RCCA?
  - RQ3: Can ChangeRCA reduce **failure identification time**?
  - RQ4: How do **RCA approaches affect** ChangeRCA?
- Metrics
  - **Hit Rate of Top-k ($HR@k$)**: probability that defective change is within top-k results
  - **Exam Score (ES):** average count of false-positive changes
- Baselines
  - SCWarn (FSE'21)
  - Gandalf (NSDI'20)
  - FUNNEL (CONEXT'15)

Experiment

- RQ1: How effective is ChangeRCA in RCCA?

Data	Approach	HR@1	⇑HR@1(%)	HR@3	⇑HR@3(%)	HR@5	⇑HR@5(%)	ES	⇓ES(%)
$\mathcal{A}$	**ChangeRCA**	**83.33**	-	**90.00**	-	**93.33**	-	**1.83**	-
	FUNNEL [60]	56.67	**47.04**	73.33	**22.73**	76.67	**21.72**	3.96	**53.78**
	SCWarn [63]	70.00	19.04	80.00	12.5	80.00	16.66	3.43	46.64
	Gandalf [24]	66.67	24.98	76.67	17.38	80.00	16.66	3.6	49.16
	w/o graph	56.67	47.04	73.33	22.73	76.67	21.72	3.96	53.78
	w/o $Score_K$	73.33	13.64	80.00	12.5	90.00	3.7	2.63	30.42
	w/o $Score_D$	70.00	19.04	80.00	12.5	80.00	16.66	3.43	46.65
	w/o $Score_T$	60.00	38.89	83.33	8.0	86.67	7.68	2.73	32.97
$\mathcal{B}$	**ChangeRCA**	**88.23**	-	**100.0**	-	**100.0**	-	**1.13**	-
	FUNNEL [60]	58.82	50	58.82	**70**	58.82	**70**	4.70	75.06
	SCWarn [63]	60.78	45.16	60.78	64.53	62.74	59.39	4.47	67.06
	Gandalf [24]	64.70	36.67	64.70	54.56	72.54	37.85	4.16	62.95
	w/o graph	37.25	**136.86**	58.82	70	58.82	70	4.92	**77.03**
	w/o $Score_K$	72.54	21.63	98.03	2.04	100.0	0	1.39	18.71
	w/o $Score_D$	80.39	9.75	98.03	2.04	100.0	0	1.27	11.02
	w/o $Score_T$	82.35	7.14	96.07	4.09	100.0	0	1.31	13.74

ChangeRCA achieves at least 19% improvement in HR@1

Experiment

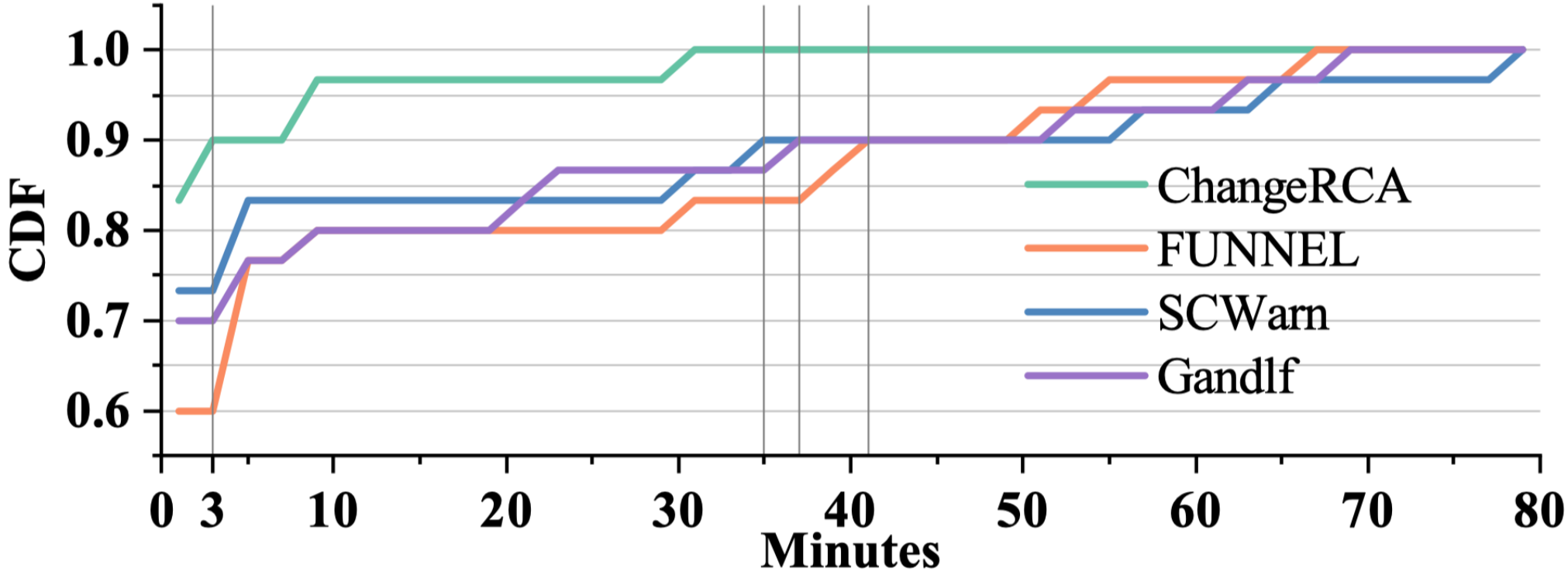
- RQ2: How useful are different scorers for RCCA?

Data	Approach	HR@1	⇑HR@1(%)	HR@3	⇑HR@3(%)	HR@5	⇑HR@5(%)
$\mathcal{A}$	**ChangeRCA**	**83.33**	-	**90.00**	-	**93.33**	-
	FUNNEL [60]	56.67	**47.04**	73.33	**22.73**	76.67	**21.72**
	SCWarn [63]	70.00	19.04	80.00	12.5	80.00	16.66
	Gandalf [24]	66.67	24.98	76.67	17.38	80.00	16.66
	w/o graph	56.67	47.04	73.33	22.73	76.67	21.72
	w/o $Score_K$	73.33	13.64	80.00	12.5	90.00	3.7
	w/o $Score_D$	70.00	19.04	80.00	12.5	80.00	16.66
	w/o $Score_T$	60.00	38.89	83.33	8.0	86.67	7.68
$\mathcal{B}$	**ChangeRCA**	**88.23**	-	**100.0**	-	**100.0**	-
	FUNNEL [60]	58.82	50	58.82	**70**	58.82	**70**
	SCWarn [63]	60.78	45.16	60.78	64.53	62.74	59.39
	Gandalf [24]	64.70	36.67	64.70	54.56	72.54	37.85
	w/o graph	37.25	**136.86**	58.82	70	58.82	70
	w/o $Score_K$	72.54	21.63	98.03	2.04	100.0	0
	w/o $Score_D$	80.39	9.75	98.03	2.04	100.0	0
	w/o $Score_T$	82.35	7.14	96.07	4.09	100.0	0

Each scorer contributes to effectiveness of ChangeRCA

Experiment

- RQ3: Can ChangeRCA reduce failure identification time?



ChangeRCA can localize 90% of defective changes in less than 3 minutes, compared to 35, 37, and 41 minutes for SCWarn, Gandalf, and FUNNEL

Experiment

- RQ4: How do RCA approaches affect ChangeRCA?

RCA Approach	HR@1(%)	HR@3(%)	HR@5(%)	ES
GIED [17]	**88.23**	**100.0**	**100.0**	**1.13**
Microscope [26]	74.50	96.07	98.03	1.49
MicroRCA [47]	88.23	96.07	98.03	1.37

Different RCA approaches have a small impact on ChangeRCA

Conclusion

- Software change **failures are common** in online systems
  - More than **40% of incidents** are directly correlated to software changes[1-2]

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*

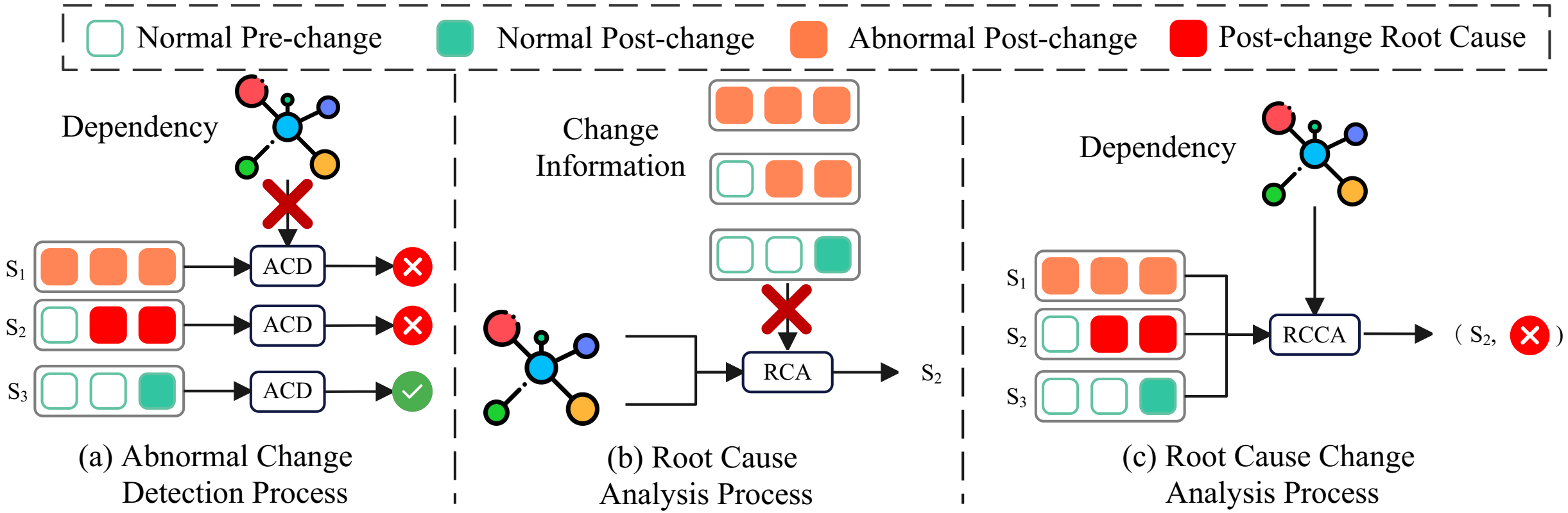
Conclusion

- Software change failure is common in online systems
  - More than 40% of incidents are directly correlated to software changes[1-2]

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*

RCCA Approach

- **Root cause change analysis (RCCA):** identify defective changes responsible for failures
  - Consider both **service dependency and change process information**

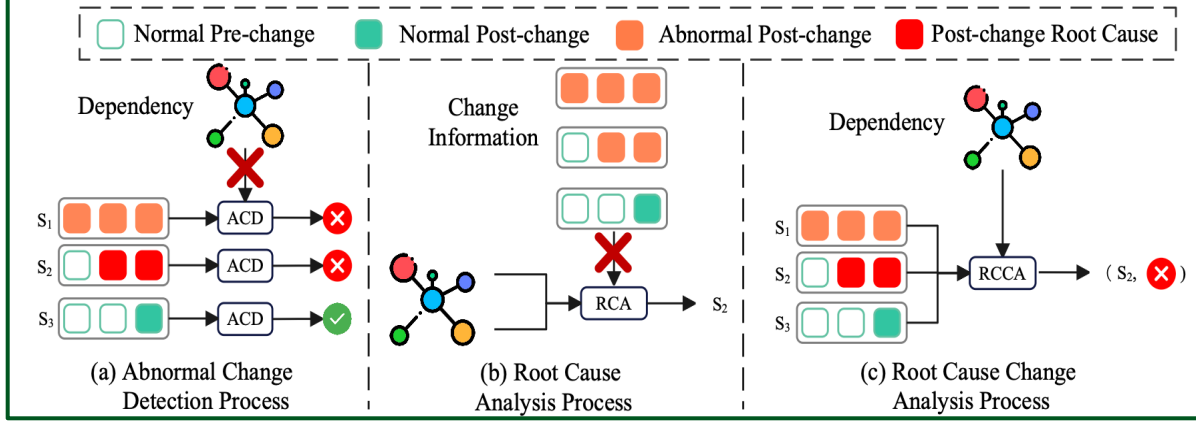


Conclusion

- Software change failure is common in online systems
  - More than 40% of incidents are directly correlated to software changes[1-2]

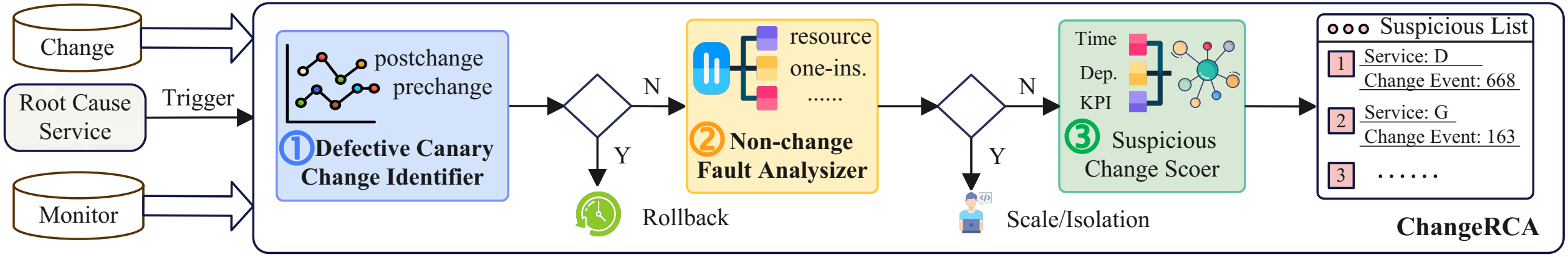
[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*

- **Root cause change analysis (RCCA):** extend the capabilities of RCA by focusing on identifying the defective changes responsible for failures
  - Both consider service dependency and change process information



Conclusion

- ChangeRCA: cascading **root cause change analysis framework** for online systems
  - Make RCCA concept applicable in practical scenarios
  - **Input**: RCA service, KPI, change process, dependency graph
  - **Output**: suspicious defective change

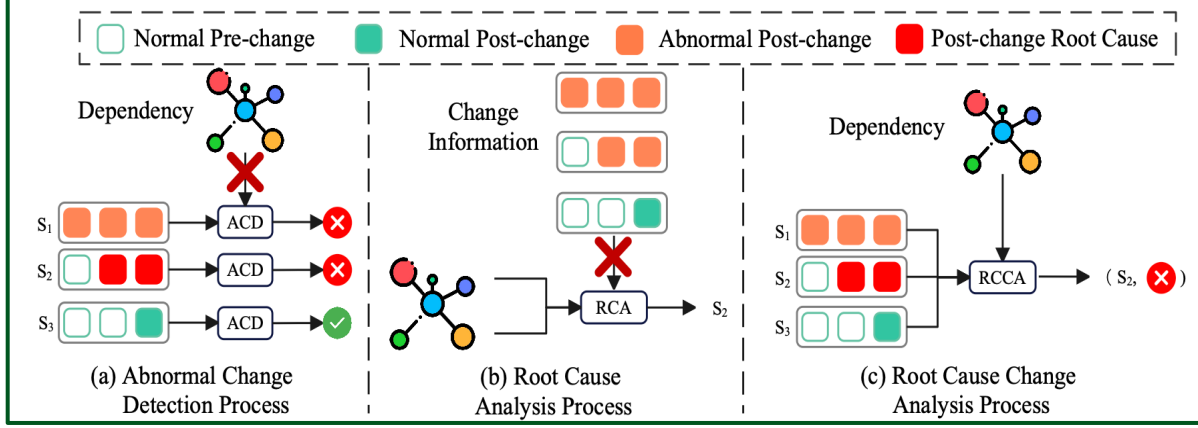


Conclusion

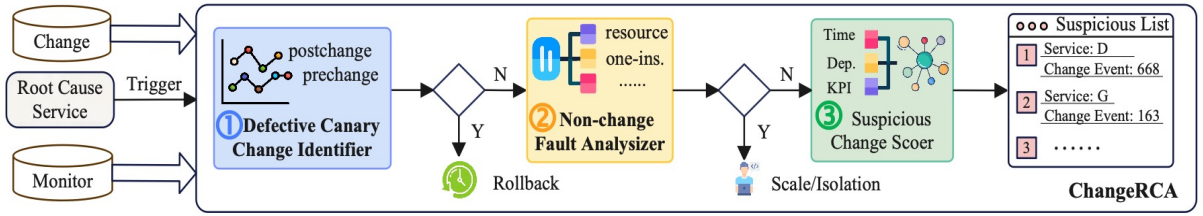
- Software change failure is common in online systems
  - More than 40% of incidents are directly correlated to software changes[1-2]

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*

- **Root cause change analysis (RCCA):** extend the capabilities of RCA by focusing on identifying the defective changes responsible for failures
  - Both consider service dependency and change process information



- ChangeRCA: cascading root cause change analysis framework for online systems
  - Make RCCA concept applicable in practical scenarios
  - Input: RCA service, KPI, change process, dependency graph
  - Output: suspicious defective change



Experiment

- RQ1: How effective is ChangeRCA in RCCA?

Data	Approach	HR@1	⇑HR@1(%)	HR@3	⇑HR@3(%)	HR@5	⇑HR@5(%)
$\mathcal{A}$	**ChangeRCA**	**83.33**	-	**90.00**	-	**93.33**	-
	FUNNEL [60]	56.67	**47.04**	73.33	**22.73**	76.67	**21.72**
	SCWarn [63]	70.00	19.04	80.00	12.5	80.00	16.66
	Gandalf [24]	66.67	24.98	76.67	17.38	80.00	16.66
	w/o graph	56.67	47.04	73.33	22.73	76.67	21.72
	w/o $Score_K$	73.33	13.64	80.00	12.5	90.00	3.7
	w/o $Score_D$	70.00	19.04	80.00	12.5	80.00	16.66
	w/o $Score_T$	60.00	38.89	83.33	8.0	86.67	7.68
$\mathcal{B}$	**ChangeRCA**	**88.23**	-	**100.0**	-	**100.0**	-
	FUNNEL [60]	58.82	50	58.82	**70**	58.82	**70**
	SCWarn [63]	60.78	45.16	60.78	64.53	62.74	59.39
	Gandalf [24]	64.70	36.67	64.70	54.56	72.54	37.85
	w/o graph	37.25	**136.86**	58.82	70	58.82	70
	w/o $Score_K$	72.54	21.63	98.03	2.04	100.0	0
	w/o $Score_D$	80.39	9.75	98.03	2.04	100.0	0
	w/o $Score_T$	82.35	7.14	96.07	4.09	100.0	0

ChangeRCA achieves at least 19% improvement in HR@1

Conclusion

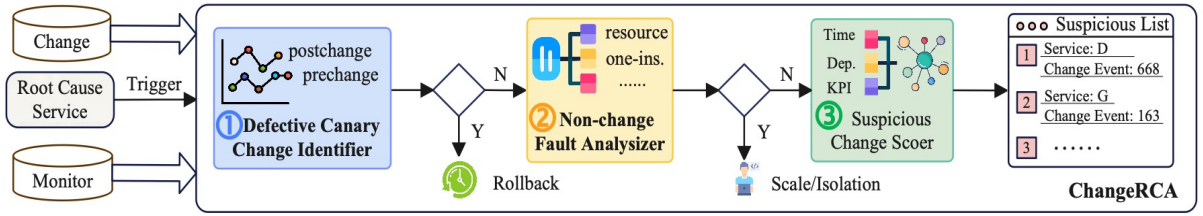
Software change failure is common in online systems

- More than 40% of incidents are directly correlated to software changes[1-2]

[1] *How to fight production incidents: an empirical study on a large-scale cloud service [SoCC'22]*
[2] *Going through the Life Cycle of Faults in Clouds: Guidelines on Fault Handling [ISSRE'22]*

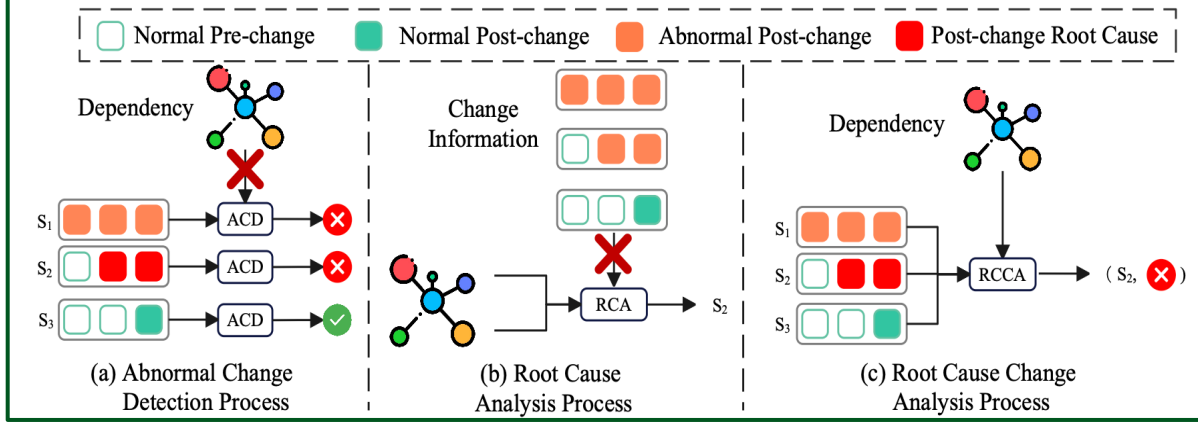
ChangeRCA: cascading root cause change analysis framework for online systems

- Make RCCA concept applicable in practical scenarios
- Input: RCA service, KPI, change process, dependency graph
- Output: suspicious defective change



Root cause change analysis (RCCA): extend the capabilities of RCA by focusing on identifying the defective changes responsible for failures

- Both consider service dependency and change process information



RQ1: How effective is ChangeRCA in RCCA?

Data	Approach	HR@1	⇑HR@1(%)	HR@3	⇑HR@3(%)	HR@5	⇑HR@5(%)
$\mathcal{A}$	**ChangeRCA**	**83.33**	-	**90.00**	-	**93.33**	-
	FUNNEL [60]	56.67	**47.04**	73.33	**22.73**	76.67	**21.72**
	SCWarn [63]	70.00	19.04	80.00	12.5	80.00	16.66
	Gandalf [24]	66.67	24.98	76.67	17.38	80.00	16.66
	w/o graph	56.67	47.04	73.33	22.73	76.67	21.72
	w/o $Score_K$	73.33	13.64	80.00	12.5	90.00	3.7
	w/o $Score_D$	70.00	19.04	80.00	12.5	80.00	16.66
	w/o $Score_T$	60.00	38.89	83.33	8.0	86.67	7.68
$\mathcal{B}$	**ChangeRCA**	**88.23**	-	**100.0**	-	**100.0**	-
	FUNNEL [60]	58.82	50	58.82	**70**	58.82	**70**
	SCWarn [63]	60.78	45.16	60.78	64.53	62.74	59.39
	Gandalf [24]	64.70	36.67	64.70	54.56	72.54	37.85
	w/o graph	37.25	**136.86**	58.82	70	58.82	70
	w/o $Score_K$	72.54	21.63	98.03	2.04	100.0	0
	w/o $Score_D$	80.39	9.75	98.03	2.04	100.0	0
	w/o $Score_T$	82.35	7.14	96.07	4.09	100.0	0

ChangeRCA achieves at least 19% improvement in HR@1

Thank you !

Q & A

Homepage

WeChat Official Accounts

https://yuxiaoba.github.io

yugb5@mail3.sysu.edu.cn

https://github.com/IntelligentDDS/ChangeRCA