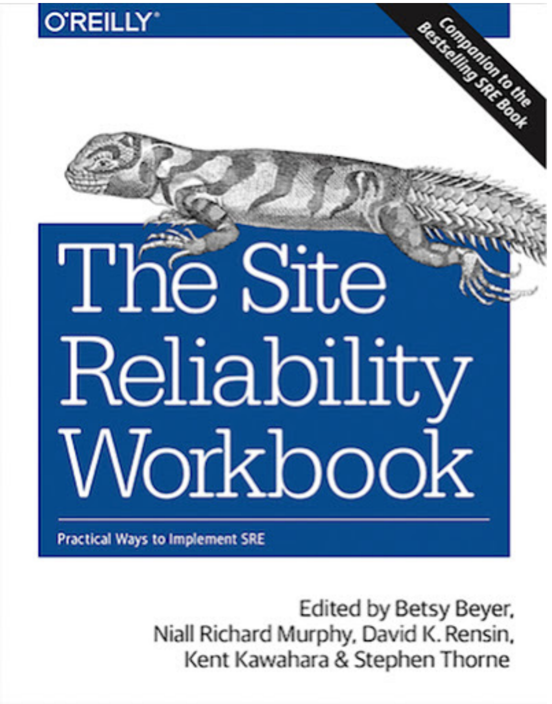
Identifying Bad Software Changes via Multimodal Anomaly Detection for Online Service Systems

Nengwen Zhao, Junjie Chen, Zhaoyang Yu, Honglin Wang, Jiesong Li, Bin Qiu, Hongyu Xu, Wenchi Zhang, Kaixin Sui, and Dan Pei

ESEC/FSE 2021

Software Change

- Software changes are frequent in online service systems
  - ✓ fix bugs
  - ✓ deploy new features
  - ✓ adapt to environmental change
  - ✓ improve software performance
  - ✓ …

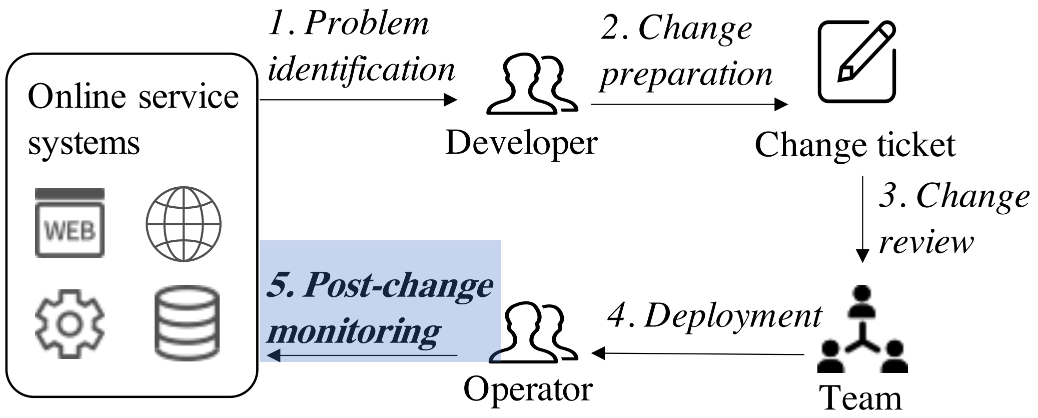


Google SRE has found that roughly 70% of outages are due to changes in a live system

Because of importing new code or configurations, software changes are more likely to incur service outages

Software Change Management

- Before deployment: risk analysis and impact assessment
- During deployment: reliable launching strategy
- After deployment: monitoring performance and identifying bad changes



Although each software change must be reviewed and tested before deployment, errors and bugs could remain uncaught in the real production environment due to the discrepancies between testing and production environment.

Empirical Study

Understand bad software changes

RQ1

What is the percentage of incidents induced by bad software changes?

RQ2

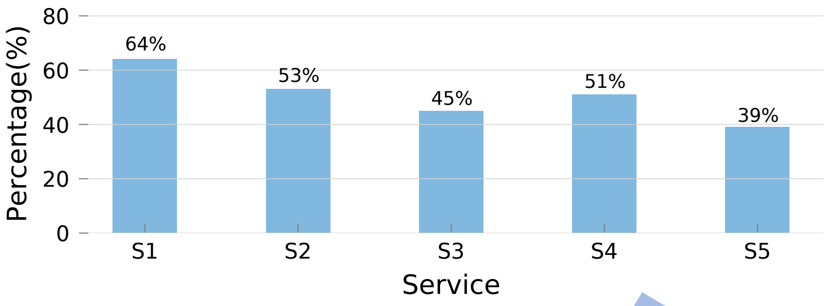
What are the root causes and behaviors of bad changes?

RQ3

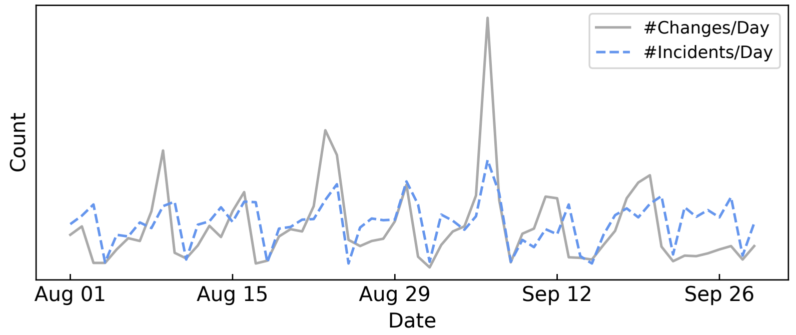
Is the current practice of identifying bad software changes good enough?

Empirical Study-1

RQ1: What is the percentage of incidents induced by bad software changes?



#Changes/day and #Incidents/day

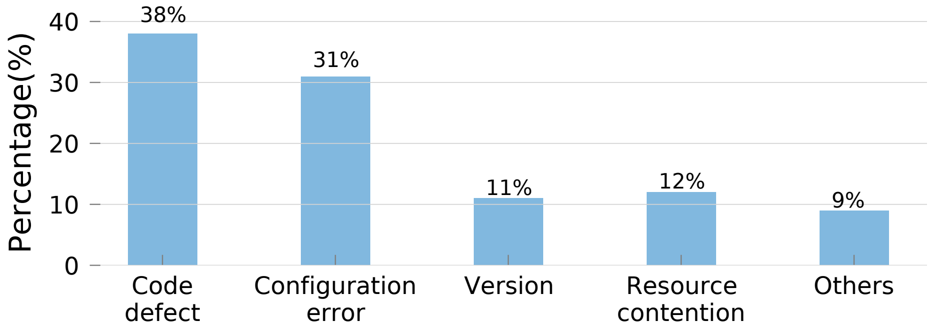


In summary, software changes are indeed failure-prone, which could bring great trouble for engineers and customers in software maintenance

Empirical Study-2

RQ2: What are the root causes and behaviors of bad changes?

Percentage of different root causes



- Change type
  - Code change (e.g., adding new features)
  - Configuration change
  - Infrastructure-layer change (e.g., replacing hardware devices)

Empirical Study-2

RQ2: What are the root causes and behaviors of bad changes?

Type	Change operation	Abnormal behaviors of multi-source monitoring data
Bad software change	Configuration error - Wrong IP address	Error messaegs in network logs (e.g., "address conflicted"); related machine KPIs and business KPIs behave abnormally
	Configuration error - Missing modification of correlated configuration [49]	Error messages in application logs; business KPIs behave abnormally
	Configuration error - Deleting white list by mistake	Business KPIs behave abnormally
	Code performance - Slow SQL, full table scan and some related database problems	Database (e.g., active session, lock wait) and related machine KPIs (e.g., disk space, CPU usage) behave abnormally; response time increases
	Code performance - Memory leak	"FullGC" log pattern appears frequently in GC log; machine KPIs (e.g., JVM heap space, memory usage) behave abnormally; response time increases
	Code performance - Code self-loop or dead loop	System load, CPU usage and other machine KPIs behave abnormally; response time increases
	Code logic bug - Wrong database table name; error date format	Error messages in application logs; success rate decreases
	Resource contention	Related machine KPIs (e.g., I/O wait, CPU usage) behave abnormally
Expected software change	Replace high-performance server; Resource expansion [45]	Related machine KPIs (e.g., CPU usage, memory usage) decrease; response time decreases
	Traffic switch	CPU usage decreases
	Code logic changes (e.g., some new steps are added to transaction process)	Related business KPIs behave abnormally (e.g., response time increases)

- Various monitoring data (business KPIs, machine KPIs and logs) from multiple sources could be influenced by software changes.
- Abnormal data behavior does not necessarily mean that this software change is bad.

Empirical Study-3

RQ3: Is the current practice of identifying bad software changes good enough?

Business KPIs

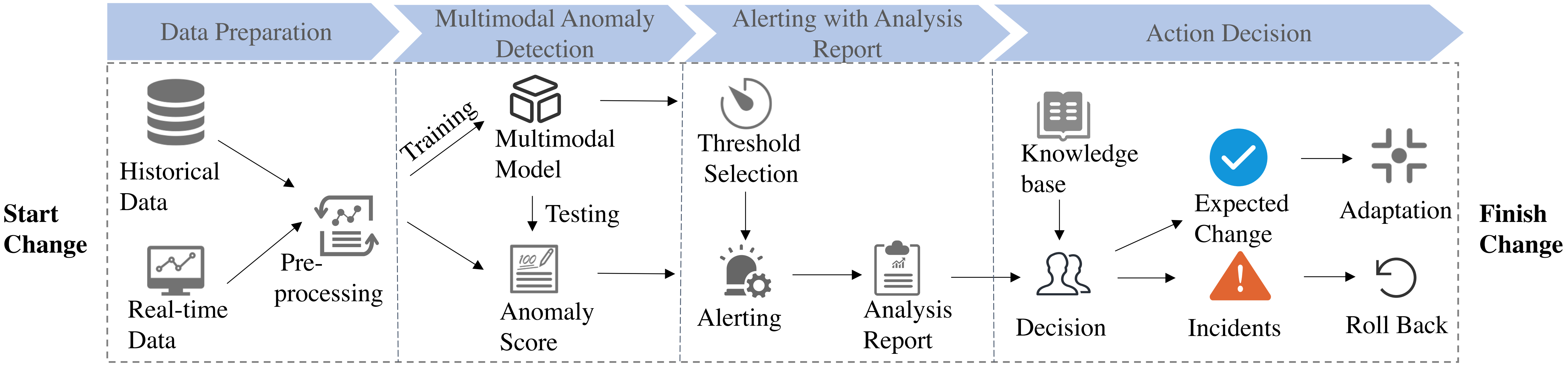
Simple anomaly detection methods

Bad or not?

- Software changes involve multi-source data
- Directly applying it ignores the specific scenario and characteristics of software change
- The drawback of the anomaly detection algorithm itself

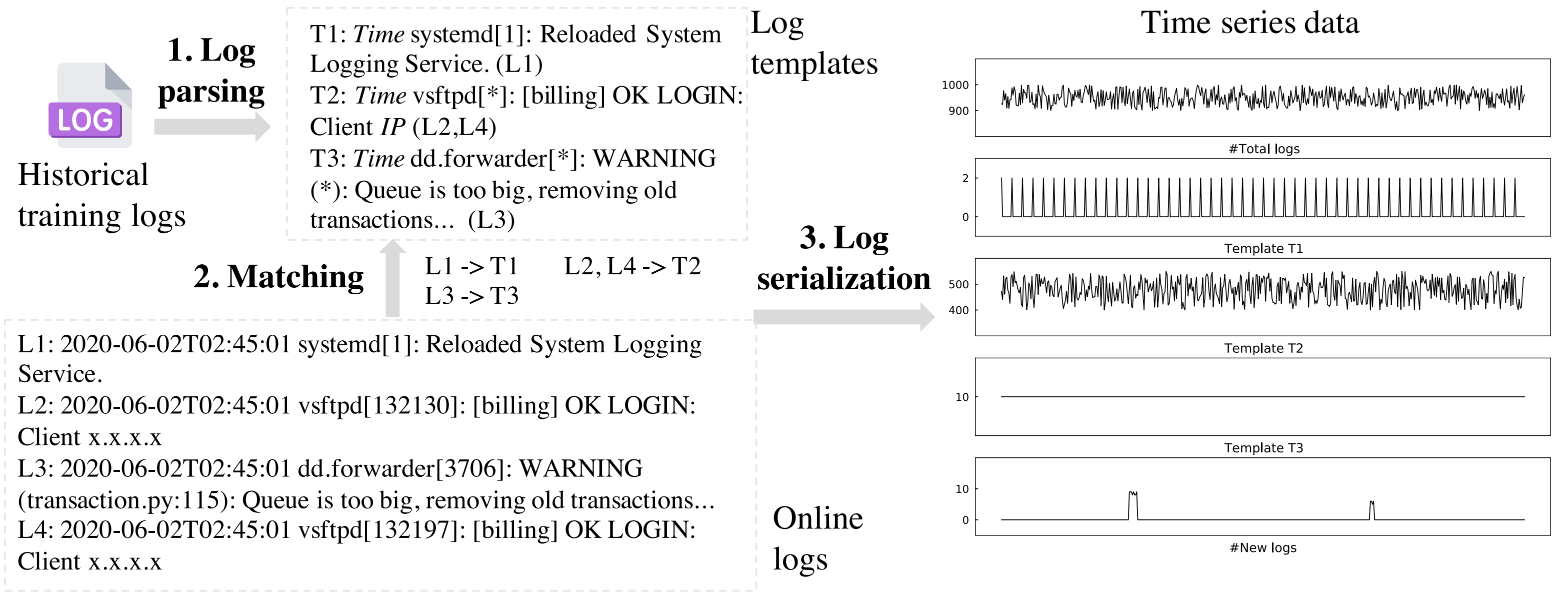
The current practice of identifying bad changes should be further improved, and it is essential to propose an effective approach to tackle the above drawbacks.

Approach



Overview of SCWarn

Data Preparation

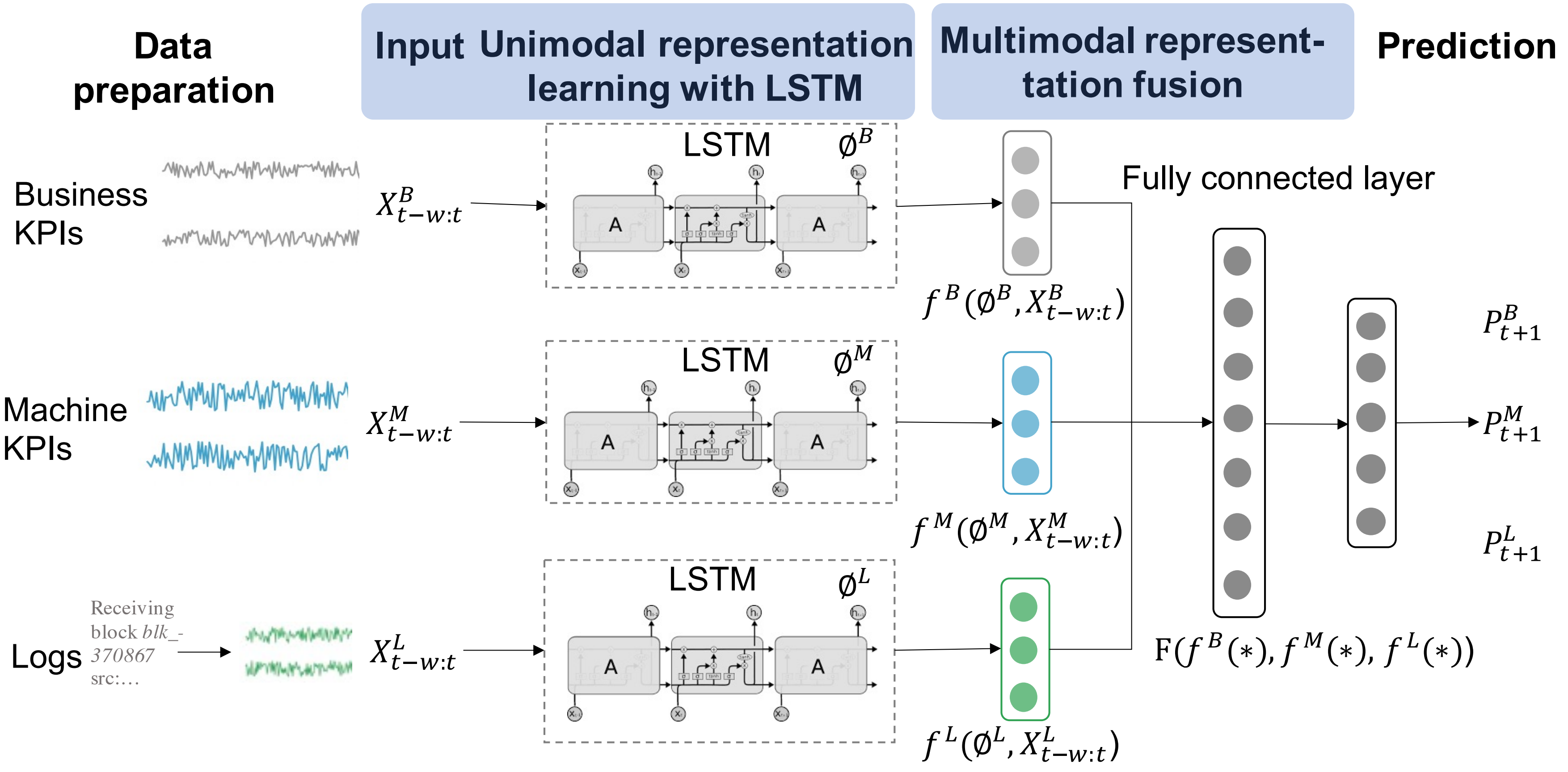


Log data preprocessing

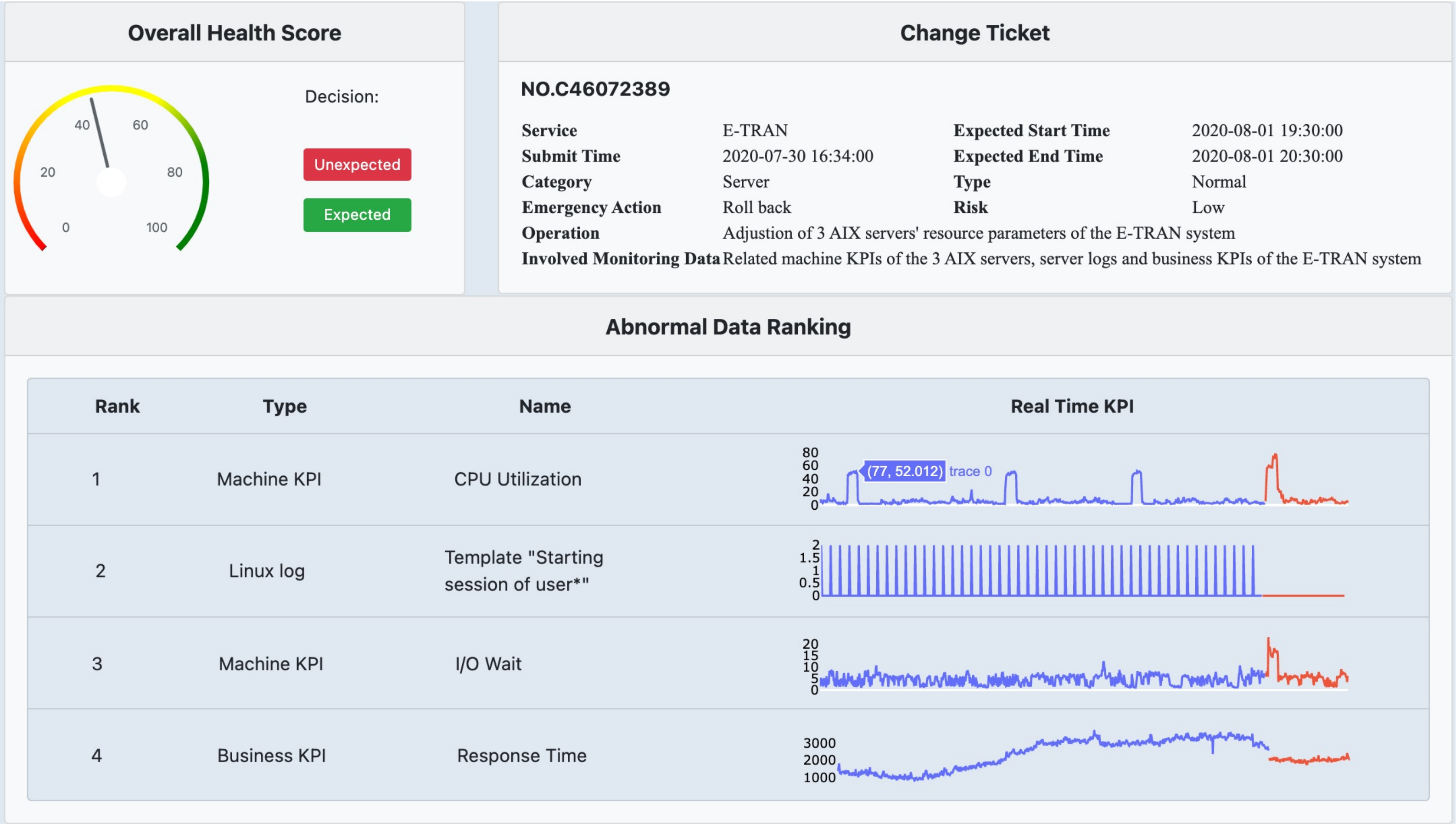
Multi-source data
- Business KPIs
- Machine KPIs
- Logs

Time series format

Multimodal Anomaly Detection



Alerting with Analysis Report



Provide a global view of the software change and help inspect related data conveniently

Evaluation

- Two benchmark systems:
  - Train-ticket
  - E-commerce
- Metrics
  - Precision, recall, F1-score, MTTD (mean time to detect)

Types and descriptions of bad software changes we injected on the benchmark systems for evaluation

Failure type	**Description**
Code defect	*F1* - Create large Java objects in program, leading to frequent fullGC and OutOfMemory error
	F2 - Inject delay into program to simulate code performance issue
	F3 - SQL statement defect leading to slow query
Configuration error	*F4* - Invalid paths which will be opened or executed
	F5 - Unsuitable size of JVM heap memory
	F6 - Database port error
	F7 - Limited number of database connections
	F8 - Non-existent database table
Software version	*F9* - Incompatible software version
Resource contention	*F10* - CPU contention

Performance

	Dataset $\mathcal{A}$				Dataset $\mathcal{B}$			
Approaches	P	R	F1	MTTD	P	R	F1	MTTD
SCWarn	**0.91**	**0.95**	**0.93**	**5.1**	**0.97**	**0.98**	**0.97**	**2.3**
Gandalf–AD	0.68	0.95	0.79	6.2	0.77	0.99	0.87	3.1
Funnel	0.77	0.69	0.73	14.0	0.76	0.87	0.81	6.4
Lumos	0.66	0.94	0.78	10.0	0.77	0.93	0.82	10.0
mFunnel	0.69	0.93	0.79	9.0	0.82	0.79	0.80	3.0
mLumos	0.85	0.83	0.84	10.0	0.72	0.99	0.83	10.0

		Dataset $\mathcal{A}$				Dataset $\mathcal{B}$			
Data source	Approaches	P	R	F1	MTTD	P	R	F1	MTTD
Business KPIs	Donut	0.65	0.92	0.76	7.3	0.94	0.75	0.83	5.4
	B-LSTM	0.86	0.75	0.80	8.2	0.99	0.88	0.93	6.6
Machine KPIs	LSTM-NDT	0.80	0.71	0.76	5.2	0.85	0.83	0.86	3.5
	OmniAnomaly	0.71	0.99	0.83	5.4	0.88	0.87	0.87	3.2
Logs	DeepLog	0.57	0.96	0.71	11.2	0.55	0.83	0.66	8.9
Multi-source data	M-AE	0.79	0.85	0.81	5.1	0.82	0.90	0.86	2.6
	M-LSTM	0.80	0.95	0.87	5.3	0.99	0.94	0.96	3.2
	Multimodal AE	0.94	0.83	0.88	6.1	0.95	0.93	0.94	4.0
	Multimodal LSTM	**0.91**	**0.95**	**0.93**	**5.1**	**0.97**	**0.98**	**0.97**	**2.3**

Our approach is indeed able to identify bad software changes accurately and timely, outperforming baseline methods and related anomaly detection methods.

Conclusion

Software change is frequent but failure-prone.

To better understand bad software changes, we conduct the first empirical study based on large-scale real-world data.

Inspired by the findings obtained from empirical study, we propose a novel approach named SCWarn to identifying bad changes accurately and timely.

An extensive study including various bad software changes confirms the effectiveness.

Thank you!

znw17@mails.tsinghua.edu.cn