2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) | 978-1-6654-1693-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/DSN53405.2022.00047

Characterizing and Mitigating Anti-patterns of Alerts in Industrial Cloud Systems

Tianyi Yang*, Jiacheng Shen*, Yuxin Su†, Xiaoxue Ren*, Yongqiang Yang‡, and Michael R. Lyu*
*Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, China.
Email: {tyyang, jcshen, lyu}@cse.cuhk.edu.hk; xiaoxueren@cuhk.edu.hk
†School of Software Engineering, Sun Yat-Sen Univeristy, Zhuhai, China. Email: suyx35@mail.sysu.edu.cn
‡Computing and Networking Innovation Lab, Cloud BU, Huawei, Shenzhen, China. Email: yangyongqiang@huawei.com

***Abstract*—Alerts are crucial for requesting prompt human intervention upon cloud anomalies. The quality of alerts significantly affects the cloud reliability and the cloud provider's business revenue. In practice, we observe on-call engineers being hindered from quickly locating and fixing faulty cloud services because of the vast existence of misleading, non-informative, non-actionable alerts. We call the ineffectiveness of alerts "anti-patterns of alerts". To better understand the anti-patterns of alerts and provide actionable measures to mitigate anti-patterns, in this paper, we conduct the first empirical study on the practices of mitigating anti-patterns of alerts in an industrial cloud system. We study the alert strategies and the alert processing procedure at Huawei Cloud, a leading cloud provider. Our study combines the quantitative analysis of millions of alerts in two years and a survey with eighteen experienced engineers. As a result, we summarized four individual anti-patterns and two collective anti-patterns of alerts. We also summarize four current reactions to mitigate the anti-patterns of alerts, and the general preventative guidelines for the configuration of alert strategy. Lastly, we propose to explore the automatic evaluation of the Quality of Alerts (QoA), including the indicativeness, precision, and handleability of alerts, as a future research direction that assists in the automatic detection of alerts' anti-patterns. The findings of our study are valuable for optimizing cloud monitoring systems and improving the reliability of cloud services.**

***Index Terms*—alert antipatterns, alert strategy, alert governance, cloud reliability, software maintenance**

I. Introduction

The boost of cloud adoption puts forward higher requirements on the reliability and availability of cloud services. Typically, cloud services are organized and managed as microservices that interact with each other and serve user requests as a whole. In a large-scale cloud microservice system, unplanned microservice anomalies happen from time to time. Some anomalies are transient, while others persist and require human intervention. If anomalies are not detected and mitigated timely, they may cause severe cloud failures and incidents, affect the availability of cloud services, and deteriorate user satisfaction [1]. Hence, prompt detection, human intervention, and mitigation of service anomalies are critical for the reliability of cloud services. To accomplish that, cloud service providers employ large-scale cloud monitoring systems that monitor the system state and generate alerts that require human intervention. Whenever anomalous states of services emerge, alerts will be generated to notify engineers to prevent service failures.

In a cloud system, an **alert** is a notification sent to On-Call Engineers (OCEs), of the form defined by the *alert strategy*, of a specific abnormal state of the cloud service, i.e., an **anomaly**. A severe enough alert (or a group of related alerts) can escalate to an **incident**, which, by definition, is any unplanned interruption or performance degradation of a service or product, which can lead to service shortages at all service levels [1]. An **alert strategy** defines the policy of alert generation, i.e., *when to generate an alert*, *what attributes and descriptions an alert should have*, and *to whom the alert should be sent*. Once an OCE receives an alert, the OCE will follow the corresponding predefined Standard Operating Procedure (**SOP**) to inspect the state of the cloud service and mitigate the service anomaly based on their domain knowledge. The *alert strategies* and *SOPs* are two key aspects to ensure a prompt and effective response to cloud alerts and incidents. In industrial practice, the two aspects are often considered and managed together because improperly designed alert strategies may lead to non-informative or delayed alerts, affecting the diagnosis and mitigation of the cloud alerts and incidents. We call the unified management of *alert strategies* and *SOPs* **alert governance**. Table I summarizes the terminologies used in this paper.

In industrial practice, a cloud provider usually deploys a cloud monitoring system to obtain the telemetry data that reflects the running state of their cloud services [2], [3]. Multiple monitoring techniques are employed to collect various types of telemetry data, including the performance indicators of the monitored service, the low-level resource utilization, the logs printed by the monitored service, etc. For normally functioning services, it is assumed that their states, as well as their telemetry data, will be stable. For a service that will fail soon, its telemetry data will fluctuate from the normal state [4], [5]. Hence, cloud providers typically conduct anomaly detection on the telemetry data to detect the deviation from the normal state. If an anomaly triggers an alert strategy, an alert will be generated, and the cloud monitoring system will notify OCEs according to the configuration of the alert strategy.

The configuration of alert strategies is empirical, which heavily depends on human expertise. Since different cloud services exhibit different attributes and serve different purposes, their alert strategies vary significantly. In particular, the

Yuxin Su is the corresponding author.

2158-3927/22/$31.00 ©2022 IEEE
DOI 10.1109/DSN53405.2022.00047

TABLE I
THE TERMINOLOGY ADOPTED IN THIS PAPER.

Term	Explanation
Anomaly	A deviation from the normal state of the cloud system, which will possibly trigger an alert.
Alert	A notification sent to On-Call Engineers (OCEs), of the form defined by the alert strategy, of a specific anomaly of the cloud system.
Incident	Any unplanned interruption or performance degradation of a service or product, which can lead to service shortages at all service levels [1].
Alert Strategy	The policy of alert generation, including *when to generate an alert*, *what attributes and descriptions an alert should have*, and *to whom the alert should be sent*.
SOP	A predefined Standard Operating Procedure (SOP) to inspect the state of the cloud system and mitigate the system abnormality upon receiving an alert. The operations can be conducted by OCEs or automatically.
Alert Governance	The unified management of *alert strategies* and *SOPs*.

empiricalness of alert strategies results from two aspects of cloud services. On the one hand, a cloud service's abnormal state may differ because each cloud service implements its own business logic. There is no one-fits-all rule for anomaly detection on cloud services, i.e., *when to generate an alert*. For example, network overload is a crucial anomaly for a virtual network service. However, high connection number becomes a real issue for a database service. On the other hand, the attributes of an alert that helps the manual inspection and mitigation of the abnormal state, e.g., the location information and the free-text title that describes the alert, are also service-specific and lack comprehensive guidelines. In other words, "*what attributes and descriptions an alert should have*" also depends on human expertise. For example, the title "Instance *x* is abnormal" is non-informative. In summary, the configuration of alert strategies, as a precursor step for human intervention in cloud anomalies, is an empirical procedure.

Manually-configured alert strategies are flexible but can also be ineffective (e.g., misleading, non-informative, and non-actionable) when the engineer is inexperienced or unfamiliar with the monitored cloud service. The ineffectiveness of alerts becomes anti-patterns that hinder the OCEs' diagnosis, especially for inexperienced OCEs. The anti-patterns of alerts, which we will elaborate in Section III, will frustrate OCEs and deteriorate cloud reliability in the long term.

In this paper, we conduct the first empirical study on the industrial practice of alert governance in Huawei Cloud[1]. The cloud system considered in this study consists of 11 cloud services and 192 cloud microservices. The procedure of our study includes 1) a quantitative assessment of over 4 million alerts in the time range of two years to identify the anti-patterns of alerts; 2) interviews with 18 experienced on-call engineers (OCEs) to confirm the identified anti-patterns and summarize the current practice to mitigate the identified anti-patterns. To sum up, we make the following contributions:

- We conduct the first empirical study on characterizing and mitigating anti-patterns of alerts in an industrial cloud system.
- We identify six anti-patterns of alerts in a production cloud system. Specifically, the six anti-patterns can be divided into two categories, namely individual anti-patterns and collective anti-patterns. Individual anti-patterns result from the ineffective patterns in one single alert strategy, including *Unclear Name or Description*, *Misleading Severity*, *Improper and Outdated Alert Strategy*, and *Transient and Toggling Alerts*. Collective anti-patterns are ineffective patterns that a bunch of alerts collectively exhibit, including *repeating* and *cascading alerts*.
- We summarize the current industrial practices for mitigating the anti-patterns of alerts, including postmortem reactions to mitigate the effect of anti-patterns and the preventative guidelines to avoid the anti-patterns. The postmortem reactions include *rule-based alert blocking* and *alert aggregation*, *pattern-based alert correlation analysis*, and *emerging alert detection*. We also describe three aspects of designing preventative guidelines for alert strategies according to our experience in Huawei Cloud.
- Lastly, we share our thoughts on prospective directions to achieve automatic alert governance. We propose to bridge the gap between manual alert strategies and cloud service upgrades by automatically evaluating the Quality of Alerts (QoA) in terms of *indicativeness*, *impact*, and *handleability*.

[1]Huawei Cloud is a global cloud provider and ranked fifth in Gartner's report [6] on the global market share of *Infrastructure as a Service* in 2020.

II. ALERTS FOR THE RELIABILITY OF CLOUD SERVICES

This section provides the preliminary knowledge for our study. We first generally introduce the reliability measures of cloud services, then describe the mechanism of alerting in cloud systems.

A. Reliability of Cloud Services

Cloud providers typically split various services into microservices and organize them into microservice architecture [7]. Microservices are small, independent, and loosely coupled modules that can be deployed independently [8]. Communicating through well-defined APIs, each microservice can be refactored and scaled independently and dynamically [9]. External requests are routed through and served by dozens of different microservices that rely on one another.

One of the major weaknesses of the microservice architecture is the difficulty in system maintenance [10], [11]. The highly decoupled nature of the microservice architecture makes the performance debugging, failure diagnosis, and fault localization in cloud systems more complex than ever [1],

[12]–[14]. A common pathway to tackle the difficulties in system maintenance is to 1) improve system observability [15]–[19] with logging, tracing, and performance monitoring, 2) employ proper alert strategies to detect system anomalies and send alerts [10], and 3) design effective SOPs to quickly mitigate the system abnormality before it escalates to severe failure and incidents. In practice, cloud providers usually deploy cloud monitoring systems to improve observability, detect anomalies, and generate alerts.

B. Alerts in Cloud Services

1) Necessities of Alerts: Service reliability is one of the most significant factors for cloud providers and their clients, but failures that prevent cloud services from properly functioning are inevitable [1]. In order to satisfy Service Level Agreements (SLAs) on the reliability of the target services, cloud providers need to deal with service and microservice anomalies before they escalate their effect into severe failures and incidents. Alerting is a practical way to achieve this goal. Figure 1 demonstrates the significance of alerts. By continuously monitoring cloud services via traces, logs, metrics, the monitoring system will send alerts[2] to On-Call Engineers (OCEs) upon detecting anomalous service states. With the information provided in the alerts, OCEs can judge with their domain knowledge, fix the problems, and clear the alert. As a result, unplanned failures and incidents can be avoided or quickly mitigated.

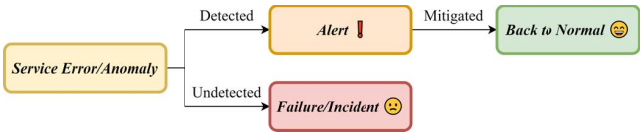


Fig. 1. The significance of alerts for cloud reliability.

2) Attributes of Alerts: Alerts have many attributes that are helpful for OCEs' diagnosis, including title of alerts, severity level, time, service name, duration, location information. The *Title of an Alert* concisely describes the alert. Typically, the title should contain information like "the affected service or microservice" and "the manifestation of the failure". The OCEs will look up the alert title to find the corresponding SOP and perform predefined actions to mitigate the alert. The *Severity Level* indicates how severe the alert is. The corresponding *Alert Strategy* defines the severity level and alert title according to the nature of the affected service or microservice. The *Time* means the time of occurrence of the alert, and *Duration* is the duration between the occurrence and the clearance of the alert. The *Location Information* contains the necessary information to locate the anomalous service or microservice. Table II shows the samples of alerts from the monitoring system of Huawei Cloud.

3) Generation of Alerts: An alert represents a specific abnormal state of the cloud system. The first and foremost step of alert generation is anomaly detection. Anomaly detection in logs [16], [20], [21], traces [11], [22], [23], and monitoring metrics [24]–[26] of the cloud system have been widely studied.

The cloud monitoring system will continuously detect anomalies and generate system reliability alerts according to the alert strategies associated with specific services or microservices. The strategies for system reliability alerts can be divided into three categories, i.e., probes, logs, and metrics.

- *Probes:* The cloud monitoring system will send probing requests to the target services and receive the heartbeat from the target services. Typically, OCEs set fixed thresholds of no-response time for different services as the strategy of probes. If a target service does not respond to the probing requests for a long time, an alert will be generated.
- *Logs:* The cloud monitoring system will process logs of the target services. OCEs can set flexible rules for different services. Typical rules of logs are keyword matching, e.g., "IF the logs contain 5 `ERROR`s in the past 2 minutes, THEN generate an alert." Traces can also be viewed as special logs and will be processed similarly.
- *Metrics:* Performance metrics are time series that show the states of a running service, e.g., latency, no. of requests, network throughput, CPU utilization, disk usage, memory utilization, etc. The alert strategy for metrics varies from static threshold to algorithmic anomaly detection.

4) Clearance of Alerts: Alerts can be cleared manually or automatically. On the one hand, after the human intervention, if the OCE confirms the mitigation of the anomaly, the OCE can manually mark the alert as "cleared". On the other hand, the cloud monitoring system can automatically clear some alerts. For system reliability alerts of *probes* and *metrics*, the cloud monitoring system will continue to monitor the status of the associated service. If the service returns to a normal state, the cloud monitoring system will mark the corresponding alert as "automatically cleared".

III. An Empirical Study on the Anti-patterns of Alerts

The research described in this paper is motivated by the pain point of alert governance in a production cloud system. In this section, we present the first empirical study of characterizing the anti-patterns of alerts[3] and how we mitigate the anti-patterns in the production cloud system. Specifically, we study the following research questions (RQs).

- **RQ1**: What anti-patterns exist in alerts? How do these anti-patterns prevent OCEs from promptly and precisely diagnosing the alert?
- **RQ2**: What is the standard procedure to process alerts? Can the standard procedure handle the anti-patterns?
- **RQ3**: What are the current **reactions** to the anti-patterns of alerts? How about their performance?
- **RQ4**: What are the current measures to **avoid** the anti-patterns of alerts? How about their performance?

[2]This paper only focuses on the alerts that indicate potential bugs and failures, i.e., the system reliability alerts.

[3]An alert always corresponds to an alert strategy. Therefore, we do not discriminate "anti-pattern of alerts" and "anti-patterns of alert strategies".

TABLE II
SAMPLE RELIABILITY ALERTS IN A CLOUD SYSTEM. THE NAMES OF MICROSERVICES ARE OMITTED DUE TO CONFIDENTIALITY.

No.	Severity	Time	Service	Alert Title	Duration	Location
1	Major	2021/05/18 06:36	Block Storage	Failed to allocate new blocks, disk full	10 min	Region=X;DC=1;...
2	Critical	2021/05/18 06:38	Database	Failed to commit changes ...	2 min	Region=X;DC=1;...
3	Critical	2021/05/18 06:39	Database	Failed to commit changes ...	5 min	Region=X;DC=1;...

To answer these research questions, we quantitatively analyzed over 4 million alerts from the production system of Huawei Cloud which serves tens of millions of users and contains hundreds of services. The time range of the alerts spans over two years. We conducted a survey involving 18 experienced OCEs to find out the current practice of mitigating the anti-patterns of alerts. Among them, 10 (55.6%) OCEs have more than 3 years of working experience. The number of OCEs with 2 to 3 years' working experience and 1 to 2 years' working experience are 3 (16.7%) and 2 (11.1%). Lastly, 3 (16.7%) OCEs' experience are less than 1 year.

A. RQ1: Anti-patterns in Alerts

Anti-patterns of alerts are misconfigured and ineffective patterns in alerts that hinder alert processing in practice. Although alerts provide essential information to OCEs for diagnosing and mitigating failures, anti-patterns of alerts hinder this process. We divide the anti-patterns into two categories, i.e., *individual anti-patterns* and *collective anti-patterns*. *Individual anti-patterns* result from the ineffectiveness of one single alert. In practice, OCEs usually have limited time to diagnose alerts. If one alert and its SOP are poorly designed, e.g., misleading steps to diagnose or non-informative description, the manual diagnosis will be difficult. *Collective anti-patterns* are ineffectiveness that alerts collectively exhibit. Sometimes, due to inappropriate configuration of alert strategy, complex dependency, and inter-influence effect in the cloud, numerous alerts may simultaneously occur. If alerts flood to OCEs or are collectively hard to handle, it will be too complicated for manual diagnosis, especially for inexperienced OCEs. Characterizing these anti-patterns is the leading step for alert governance.

For this research question, we analyzed more than 4 million alerts over two years to characterize the anti-patterns of alerts. The total number of alert strategies in this empirical study is 2010. To select the candidates of individual anti-patterns, we group the alerts according to the alert strategies, then calculate each strategies' average processing time. The alert strategies that take the top 30% longest time to process are selected as the candidates of individual anti-patterns. To find cases of collective anti-patterns, we first group all the alerts by the hour they occur and the region they belong to. Then we count the number of alerts per hour per region. If the number of alerts per hour per region exceeds 200[4], we select all the alerts in this group as the candidate of collective anti-patterns. We also went through the incident reports over the past two years to seek the ineffectiveness in alerts recorded by OCEs. We get five candidate cases of individual anti-patterns and two candidate cases of collective anti-patterns. After that, we ask two experienced OCEs to mark whether they think the candidate ineffective pattern in alerts is an anti-pattern. If they both agree, we include it as an anti-pattern. If disagreements occur, another experienced OCE is invited to examine the pattern. As a result, we summarized four individual anti-patterns and two collective anti-patterns.

[4]We set the threshold as 200 as the estimated maximum number of alerts an OCE team can deal with is 200. Experienced OCEs confirm the threshold.

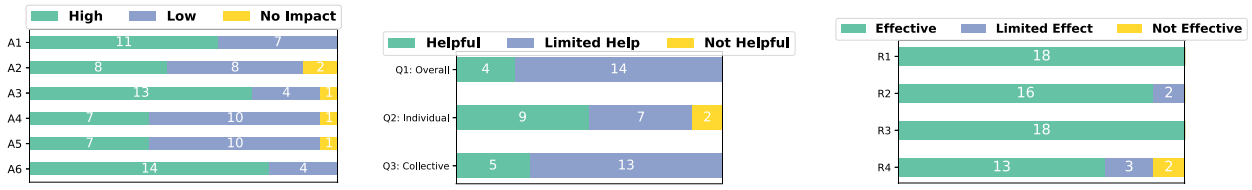
Our survey asked the OCEs to determine the impact of different anti-patterns on alert diagnosis. Figure 2(a) shows the answers' distributions. Each bar represents one anti-pattern, which is elaborated below.

1) Individual anti-patterns: Individual anti-patterns are the ineffectiveness of a single alert, including unclear name or description, misleading severity, and improper and outdated generation rule.

[A1] *Unclear Name or Description*. Unclear alert name or alert description obstructs the OCEs from gaining intuitive judgment at the first sight, which slows down the diagnosis and even hinders OCEs from knowing the logical connections from the alert to other alerts. Typical unclear alert names describe the system state in a very general way with vague words, e.g., "Elastic Computing Service is abnormal", "Instance x is abnormal", "Component y encounters exceptions", and "Computing cluster has risks". All OCEs agree with the impact of *unclear name or description*, and 61.1% of them think the impact is high.

[A2] *Misleading Severity*. Severity helps OCEs to prioritize which alert to diagnose first. Inappropriately high severity level takes up OCE's time for dealing with less essential alerts, while too low severity level may lead to missing important alerts. In our survey, 88.9% of OCEs agree with the impact of *misleading severity*. In practice, we find that the setting of severity heavily depends on domain knowledge. With the update of the cloud system, especially the enhancement of fault tolerance mechanisms, the severity may also change.

[A3] *Improper and Outdated Generation Rule*. Typically, the cloud monitoring system will continuously monitor the performance indicators of both lower-level infrastructures (e.g., CPU usage, disk usage) and higher-level services (e.g., request per second, response latency). If any indicator increases over or drops below the predefined thresholds, an alert will be generated. Although the performance indicators of lower-level infrastructures can provide valuable information when the root cause of the alert is failures of lower-level infrastructures (e.g., high CPU usage), due to the fault-tolerance techniques applied in cloud services, the performance indica-



(a) How about the impact of different anti-patterns to alert diagnosis?

(b) How helpful are the predefined SOPs?

(c) How about the effectiveness of current reactions to anti-patterns?

Fig. 2. A survey about the current practice of mitigating the anti-patterns of alerts.

tors of lower-level infrastructures do not have definite effect on the quality of cloud services from the perspective of customers. According to our survey, 72.2% of OCEs agree that the impact of *improper and outdated generation rule* is high.

[A4] *Transient and Toggling Alerts*. As mentioned in Section II-B4, the cloud monitoring system can automatically clear some alerts. When the interval between the generation time and automatic clearance time of an alarm is less than a certain value (known as the intermittent interruption threshold), the alert is called a transient alert. Commonly speaking, a transient alert is an alert that lasts for a short time. When the same alert is generated and cleared multiple times (i.e., oscillation), and the number of oscillations is greater than a certain value (known as the oscillation threshold), it is called a toggling alert. Transient and toggling alerts are usually caused by alert strategies being too sensitive to the fluctuation of the metrics. Transient and toggling alerts cause fatigue of OCEs and also distract the OCEs from being dealing with other important alerts. Although there are disagreements on the level of impact, most OCEs (94.4%) think the impact exists.

2) Collective anti-patterns: Collective anti-patterns result from the ineffective patterns of a bunch of alerts that occur in a short time scope. Zhao et al. [10] defined numerous alerts (e.g., hundreds of alerts) from different cloud services in a short time (e.g., one minute) as "alert storm", and conducted several case studies of alert storms. In alert storms, even if all the individual alerts are effective, the large number of alerts may still set obstacles for OCEs and greatly affect the system reliability in the following three ways. Firstly, during an alert storm, many alerts are generated. If OCEs check each alert manually, the troubleshooting will take unacceptably long time. Secondly, since alert storms occur frequently [10], the OCEs will continually receive alerts by email, SMS, or even phone call. According to our study, alert storms occur weekly or even daily, and 17 out of 18 interviewed OCEs say that the alert storms greatly fatigue them. Lastly, the overwhelming number of alerts adds pressure to the monitoring system, so the latency of generating new alerts may increase.

Inspired by [10], we summarize the following collective anti-patterns from confirmed cases of alert storms in Huawei Cloud. In this study, if the number of alerts from a region exceeds 100 in an hour, we count it as an alert storm. Consecutive hours of alert storm will be merged into one. Among the two collective anti-patterns, "cascading alerts" has already been observed by [10], but "repeating alerts" has not. In particular, we demonstrate the collective anti-patterns of alerts with a representative alert storm that happened from 7:00 AM to 11:59 AM in Huawei Cloud. During the alert storm, totally 2751 alerts were generated, among which we observeed both collective anti-patterns as described below.

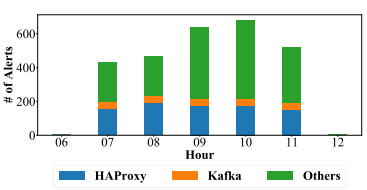


Fig. 3. Repeating alerts in an alert storm.

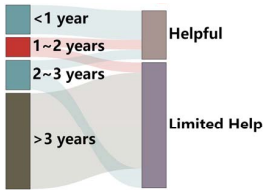


Fig. 4. Answers to Q1 "Overall Helpfulness" regarding OCEs' working experience.

[A5] *Repeating Alerts*. Repeating alerts means that alerts from the same alert strategy appear repeatedly. Sometimes the repeated alerts may last for several hours. This is usually due to the inappropriate frequency of alert generation. For example, in Figure III-A2, we count the number of alerts per strategy. The total number of alerts is 2751, and the number of effective alert strategies is 200. To make the figure clear, we only show the name of the top two alerts. All other alerts are classified as "Others" in the figure. The alert "haproxy process number warning", abbreviated as `HAProxy` in the figure, takes up around 30% of the total number of alerts in each hour. However, it is only a `WARNING` level alert, i.e., the lowest level. Even though an individual alert is straightforward to process, it is still time-consuming to deal with it when it occurs repeatedly. If one rule continually generates alerts, it will distract OCEs from dealing with the more essential alerts. Most OCEs (94.4%) agree with the impact of *repeating alerts*.

[A6] *Cascading Alerts*. Modern cloud systems are composed of many microservices that depend on each other [22]. When a service enters an anomalous state, other services that rely on it will probably suffer from anomalous states as well. Such anomalous states can propagate through the service-calling structure [27]. Despite various fault tolerance mechanisms being introduced, minor anomalies are still common to magnify their impact and eventually affect the entire system. Each of the affected services will generate many anomalous monitoring metrics, resulting in many alerts (e.g., thousands of alerts per hour). As a consequence, the alerts burst and flood to the OCEs. Although the alerts are different, they are implicitly related because they originate from the cascading effect of one single failure. Manually inspecting the alerts is hard without sufficient knowledge of the dependencies in the cloud system.

All interviewed OCEs agree with the impact of *cascading alerts*. Table II shows a simplified sample of cascading alerts. By manually inspecting the alerts, experienced OCEs would infer that the alert 1 possibly cause alert 2 because 1) Alert 2&3 occurred right after alert 1 and 2) The relational database service relies on the block storage service as the backend. If the relational database service failed to commit changes, i.e., write data, one possible reason is that the storage service failed.

Finding 1: Individual anti-patterns and collective anti-patterns widely exist. They hinder alert diagnosis to different extent.

B. RQ2: Standard Alert Processing Procedure

SOP for alert `nginx_cpu_usage_over_80`	
Description	CPU usage of nginx instance is higher than 80%
Generation Rule	Continuously check the CPU usage of nginx instance, generate the alert when usage is higher than 80%.
Potential Impact	Affects the forwarding of all requests.
Possible Causes	a) The workload is too high. b) .....
Steps to Diagnose	Step 1: execute command `top -bn1` in the instance. Step 2: .....

Fig. 5. An example Standard Operation Procedure.

The Standard Operation Procedure (SOP) defines the procedure to process a single alert. For each alert, its SOP includes the alert name, the alert description, the generation rule of the alert (i.e., alert strategy), the potential impact on the cloud system, the possible causes, and the steps to process the alert. Figure 5 shows an example SOP of the alert `nginx_cpu_usage_over_80`. The OCEs can follow the SOP to process the alert upon receiving the alert. According to our survey, only 22.2% of OCEs think current SOPs are helpful (Q1, Figure 2(b)), and the other 77.8% of OCEs say the help is limited. The SOPs are deemed to show limited help by all OCEs with over 3 years' experience, taking up 71.4% of all OCEs selected "Limited Help" for Q1 (Figure III-A2). Moreover, SOPs are considered much less helpful for diagnosing collective anti-patterns (Q3, Figure 2(b)) than individual anti-patterns (Q2, Figure 2(b)).

Finding 2: SOPs can help OCEs quickly process alerts, but the help is limited. SOPs are considered less helpful when dealing with collective anti-patterns.

C. RQ3: Reactions to Anti-patterns

Depending on the number of alerts, OCEs react differently. When the number of alerts is relatively small, OCEs will scan through all the reported alerts. Then they will manually rule out alerts that are not of great importance and deal with critical alerts that will affect the whole system.

OCEs react differently when the number of alerts becomes too large. According to our interview with senior OCEs in Huawei Cloud, they typically take four kinds of reactions, i.e., alert blocking, alert aggregation, alert correlation analysis, and emerging alert detection. In practice, we observe that although the reactions are considered effective, they need to be reconfigured after the update of cloud services or alert strategies.

[R1] *Alert Blocking*. When OCEs find that transient alerts, toggling alerts, and repeating alerts provide no information about service anomaly, they can treat these alerts as noise and block them with alert blocking rules. As a result, these non-informative alerts will not distract OCEs from quickly identifying the root causes of service anomalies.

[R2] *Alert Aggregation*. When dealing with large amounts of alerts, there may be many duplicate alerts in a time period. For the non-informative alerts, OCEs will employ alert blocking introduced before to facilitate analysis. For the informative ones, they will adopt alert aggregation. To be more specific, OCEs will set rules to aggregate alerts in a period and use the number of alerts as another feature [28]. By doing so, OCEs can quickly identify critical alerts and focus more on the information provided by them.

[R3] *Alert Correlation Analysis*. Apart from the information provided by the alerts and their statistical characteristics, OCEs will also leverage other exogenous information to analyze the correlation of alerts. Two kinds of exogenous information are used to correlate alerts. The first is the dependencies of alert strategies, which indicate the spread of alerts in the cloud services [29]. For instance, if a source alert triggers another alert, OCEs will be more interested in the source alert, potentially the root cause of future service failures. They will associate all the derived alerts with their source alerts and diagnose the source alerts only. Another exogenous information is the topology of cloud services. Based on the topology of services, OCEs will set rules to correlate alerts based on the services that generated them. With this kind of correlation, OCEs can quickly pinpoint the root cause of a large number of alerts by following the topological correlation.

[R4] *Emerging Alert Detection*. Due to the large scale of cloud services, manually configured dependencies of alert strategies could not cover all the alert strategies. This may lead to the failure of alert correlation analysis. For example, a few alerts corresponding to a root cause (i.e., emerging alerts) appear first. If they are not dealt with seriously, when the root cause escalates its influence, numerous cascading alerts will be generated. The lack of critical association rules will prevent the OCEs from discovering the correlation and quickly alert diagnosis. This usually happens on gray failures like memory leak and CPU overloading. Hence, it would be helpful to capture the implicit dependencies. We employ the adaptive online Latent Dirichlet Allocation [30], [31] to capture the implicit dependencies. OCEs could detect these emerging alerts as early as possible for faster alert diagnosis with the implicit dependencies.

Figure 2(c) shows OCEs' opinions about the effectiveness of the four reactions. In general, the effectiveness of all four reactions is relatively high.

Finding 3: Current reactions are considered effective, but the configurations of such reactions still require domain knowledge.

D. RQ4: Avoidance of Anti-patterns

To avoid the alert anti-patterns from occurring, Huawei Cloud also adopts preventative guidelines and conducts periodical reviews on alert strategies. We summarize the generic aspects to consider when designing the guidelines. The guidelines are designed by experienced OCEs and guide from three aspects of alerts.

- *Target* means what to monitor. The performance metrics highly related to the service quality should be monitored.
- *Timing* means when to generate an alert upon the manifestation of anomalies. Sometimes an anomaly does not necessarily mean the service quality will be affected.
- *Presentation* means whether the alerts' attributes are helpful for alert diagnosis.

However, our interview with OCEs shows that the preventative guidelines are not strictly obeyed in practice. Most (88.9%) OCEs agree that strictly following the guidelines will make alert diagnosis easier.

Finding 4: The preventative guidelines could reduce the anti-patterns and assist in alert diagnosis if they are carefully designed and strictly obeyed.

IV. Future Directions

Although several postmortem reactions and preventative guidelines are adopted (Section III), according to our study, the problem of alert anti-patterns is still prevailing in industrial cloud monitoring systems because most current measures still require manual configuration. As for the alert blocking, OCEs need to inspect each alert and set rules manually. How to define the blocking rules and when to invalidate these rules become a crucial problem. A similar problem also exists in alert correlation. As for alert correlation analysis, OCEs also need to inspect alert generating rules and service topology documents apart from reading alerts, which incurs a considerable burden to OCEs. Moreover, the effectiveness of the reactions also lacks clear criteria to evaluate. OCEs can only estimate the effectiveness of the reactive measures by their feeling. Therefore, outdated reactive measures is hard to detect. As a result, the whole process of alert governance becomes time-consuming and laborious.

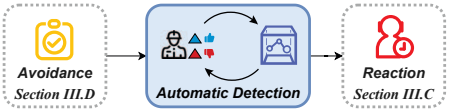


Fig. 6. Incorporating human knowledge and machine learning to detect anti-patterns of alerts.

In Figure 6, we formulate the three stages of the mitigation of alert anti-patterns. We already shared our experience of avoiding and reacting to alert anti-patterns in Section III. To close the gap between manual alert strategies and cloud system upgrades, we propose to explore the automatic detection of alert anti-patterns. Automatic evaluation of the Quality of Alerts (QoA) will be a promising approach to the automatic detection of alert anti-patterns.

Based on our empirical study, we propose three criteria to measure the quality of alerts (QoA), including *indicativeness*, *precision*, and *handleability*.

- *Indicativeness* measures whether the alert can indicate the failures that will affect the end users' experience.
- *Precision* measures whether the alert can correctly reflect the severity of the anomaly.
- *Handleability* measures whether the alert can be quickly handled. The handleability depends on the target and the presentation of the alert. Improper target or unclear presentation decreases the handleability.

In the future, incorporating human knowledge and machine learning to evaluate the three aspects of alerts deserves more exploration. In particular, OCEs provide their domain knowledge by creating labels like "high/low precision/handleability/indicativeness" for each alerts during alert processing. With the labels, a machine learning model could be trained and continuously updated so that it can automatically absorb the human knowledge for future QoA evaluation.

V. Related Work

Many works focus on processing alerts of cloud services and microservices. One of the essential tasks of alert processing is to reduce the enormous amount of reported alerts to facilitate failure diagnosis. Alert correlation [32] and clustering [10], [33], [34] are two common techniques employed to help OCEs find critical alerts and repair the system in a short period. Li et al. [35] proposes to generate incidents based on the system alerts to prevent services from future failures. Unlike all prior works, our paper focuses on not only how to deal with alerts after they are generated, but also how to generate better alerts and conduct better alert governance.

VI. Conclusion

This paper conducts the first empirical study to characterize the anti-patterns in cloud alerts. We also summarize the industrial practices of mitigating the anti-patterns by postmortem reactions and preventative guidelines. We wish our study to inspire further research on automatic QoA evaluation and anti-pattern detection and benefit the reliability of the cloud services in the long run.

Acknowledgment

The work was supported by Key-Area Research and Development Program of Guangdong Province (No. 2020B010165002), Key Program of Fundamental Research from Shenzhen Science and Technology Innovation Commission (No. JCYJ20200109113403826), and the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14210920).

REFERENCES

[1] Z. Chen, Y. Kang, L. Li, X. Zhang, H. Zhang, H. Xu, Y. Zhou, L. Yang, J. Sun, Z. Xu, Y. Dang, F. Gao, P. Zhao, B. Qiao, Q. Lin, D. Zhang, and M. R. Lyu, "Towards intelligent incident management: why we need it and how we make it," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*. ACM, 2020, pp. 1487–1497.

[2] Z. Li, Q. Cheng, K. Hsieh, Y. Dang, P. Huang, P. Singh, X. Yang, Q. Lin, Y. Wu, S. Levy, and M. Chintalapati, "Gandalf: An intelligent, end-to-end analytics service for safe deployment in large-scale cloud infrastructure," in *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*. USENIX Association, 2020, pp. 389–402.

[3] C. L. Dickson, "A working theory-of-monitoring," Google, Inc., Tech. Rep., 2013. [Online]. Available: https://www.usenix.org/conference/lisa13/working-theory-monitoring

[4] P. Huang, C. Guo, L. Zhou, J. R. Lorch, Y. Dang, M. Chintalapati, and R. Yao, "Gray failure: The achilles' heel of cloud-scale systems," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems, HotOS 2017, Whistler, BC, Canada, May 8-10, 2017*. ACM, 2017, pp. 150–155.

[5] H. Wang, Z. Wu, H. Jiang, Y. Huang, J. Wang, S. Köprü, and T. Xie, "Groot: An event-graph-based approach for root cause analysis in industrial settings," in *ASE '21: 36th IEEE/ACM International Conference on Automated Software Engineering, Virtual Event, Australia, November 15-19, 2021*. IEEE/ACM, 2021, pp. 1–12.

[6] D. Blackmore, C. Tornbohm, D. Ackerman, C. Graham, S. Matson, T. Lo, T. Singh, A. Roy, C. Tenneson, M. Sawai, E. Kim, E. Anderson, S. Nag, N. Barton, N. Sethi, R. Malik, B. Williams, C. Healey, R. Buest, T. Wu, K. Madaan, S. Sahoo, H. Singh, and P. Sullivan, "Market share: It services, worldwide, 2020," Tech. Rep., 2021.

[7] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[8] M. A. Doc, "Microservices architecture style," 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices

[9] M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil, "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *2015 10th Computing Colombian Conference (10CCC)*. IEEE, 2015, pp. 583–590.

[10] N. Zhao, J. Chen, X. Peng, H. Wang, X. Wu, Y. Zhang, Z. Chen, X. Zheng, X. Nie, G. Wang, Y. Wu, F. Zhou, W. Zhang, K. Sui, and D. Pei, "Understanding and handling alert storm for online service systems," in *ICSE-SEIP 2020: 42nd International Conference on Software Engineering, Software Engineering in Practice, Seoul, South Korea, 27 June - 19 July, 2020*. ACM, 2020, pp. 162–171.

[11] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, D. Liu, Q. Xiang, and C. He, "Latent error prediction and fault localization for microservice applications by learning from system trace logs," in *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2019, Tallinn, Estonia, August 26-30, 2019*. ACM, 2019, pp. 683–694.

[12] X. Zhang, Y. Xu, S. Qin, S. He, B. Qiao, Z. Li, H. Zhang, X. Li, Y. Dang, Q. Lin, M. Chintalapati, S. Rajmohan, and D. Zhang, "Onion: identifying incident-indicating logs for cloud systems," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*. ACM, 2021, pp. 1253–1263.

[13] X. Zhang, Q. Lin, Y. Xu, S. Qin, H. Zhang, B. Qiao, Y. Dang, X. Yang, Q. Cheng, M. Chintalapati, Y. Wu, K. Hsieh, K. Sui, X. Meng, Y. Xu, W. Zhang, F. Shen, and D. Zhang, "Cross-dataset time series anomaly detection for cloud systems," in *2019 USENIX Annual Technical Conference, USENIX ATC 2019, Renton, WA, USA, July 10-12, 2019*. USENIX Association, 2019, pp. 1063–1076.

[14] Y. Gan, Y. Zhang, K. Hu, D. Cheng, Y. He, M. Pancholi, and C. Delimitrou, "Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*. ACM, 2019, pp. 19–33.

[15] P. Huang, C. Guo, J. R. Lorch, L. Zhou, and Y. Dang, "Capturing and enhancing in situ system observability for failure detection," in *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*. USENIX Association, 2018, pp. 1–16.

[16] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Comput. Surv.*, vol. 54, no. 6, Jul. 2021. [Online]. Available: https://doi.org/10.1145/3460345

[17] A. Pecchia, M. Cinque, G. Carrozza, and D. Cotroneo, "Industry practices and event logging: assessment of a critical software development process," in *Proc. of the 37th IEEE/ACM International Conference on Software Engineering (ICSE)*, 2015, pp. 169–178.

[18] K. Yao, G. B. de Pádua, W. Shang, C. Sporea, A. Toma, and S. Sajedi, "Log4perf: suggesting and updating logging locations for web-based systems' performance monitoring," *Empir. Softw. Eng.*, vol. 25, no. 1, pp. 488–531, 2020.

[19] S. He, Q. Lin, J. Lou, H. Zhang, M. R. Lyu, and D. Zhang, "Identifying impactful service system problems via log analysis," in *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE 2018, Lake Buena Vista, FL, USA, November 04-09, 2018*. ACM, 2018, pp. 60–70.

[20] V. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *ASE '21: 36th IEEE/ACM International Conference on Automated Software Engineering, Virtual Event, Australia, November 15-19, 2021*. IEEE/ACM, 2021, pp. 1–12.

[21] N. Zhao, H. Wang, Z. Li, X. Peng, G. Wang, Z. Pan, Y. Wu, Z. Feng, X. Wen, W. Zhang, K. Sui, and D. Pei, "An empirical investigation of practical log anomaly detection for online service systems," in *ESEC/FSE '21: 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, August 23-28, 2021*. ACM, 2021, pp. 1404–1415.

[22] T. Yang, J. Shen, Y. Su, X. Ling, Y. Yang, and M. R. Lyu, "Aid: Efficient prediction of aggregated intensity of dependency in large-scale cloud systems," in *ASE '21: 36th IEEE/ACM International Conference on Automated Software Engineering, Virtual Event, Australia, November 15-19, 2021*. IEEE/ACM, 2021, pp. 1–12.

[23] X. Guo, X. Peng, H. Wang, W. Li, H. Jiang, D. Ding, T. Xie, and L. Su, "Graph-based trace analysis for microservice architecture understanding and problem diagnosis," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*. ACM, 2020, pp. 1387–1397.

[24] Y. Meng, S. Zhang, Y. Sun, R. Zhang, Z. Hu, Y. Zhang, C. Jia, Z. Wang, and D. Pei, "Localizing failure root causes in a microservice through causality inference," in *28th IEEE/ACM International Symposium on Quality of Service, IWQoS 2020, Hangzhou, China, June 15-17, 2020*. IEEE, 2020, pp. 1–10.

[25] P. Liu, Y. Chen, X. Nie, J. Zhu, S. Zhang, K. Sui, M. Zhang, and D. Pei, "Fluxrank: A widely-deployable framework to automatically localizing root cause machines for software service failure mitigation," in *30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019, Berlin, Germany, October 28-31, 2019*. IEEE, 2019, pp. 35–46.

[26] G. Zhao, S. Hassan, Y. Zou, D. Truong, and T. Corbin, "Predicting performance anomalies in software systems at run-time," *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 3, pp. 33:1–33:33, 2021.

[27] A. W. Services, "Aws well-architected framework," 2020. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html

[28] Z. Chen, J. Liu, Y. Su, H. Zhang, X. Wen, X. Ling, Y. Yang, and M. R. Lyu, "Graph-based incident aggregation for large-scale online service systems," in *ASE '21: 36th IEEE/ACM International Conference on Automated Software Engineering, Virtual Event, Australia, November 15-19, 2021*. IEEE/ACM, 2021, pp. 1–12.

[29] R. Melo and D. Macedo, "A cloud immune security model based on alert correlation and software defined network," in *28th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2019, Naples, Italy, June 12-14, 2019*. IEEE, 2019, pp. 52–57.

[30] T. Yang, C. Gao, J. Zang, D. Lo, and M. R. Lyu, "TOUR: dynamic topic and sentiment analysis of user reviews for assisting app release," in *Companion of The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*. ACM / IW3C2, 2021, pp. 708–712.

[31] C. Gao, J. Zeng, M. R. Lyu, and I. King, "Online app review analysis for identifying emerging issues," in *Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018*. ACM, 2018, pp. 48–58.

[32] S. A. Mirheidari, S. Arshad, and R. Jalili, "Alert correlation algorithms: A survey and taxonomy," in *Cyberspace Safety and Security - 5th International Symposium, CSS 2013, Zhangjiajie, China, November 13-15, 2013, Proceedings*, ser. Lecture Notes in Computer Science, vol. 8300. Springer, 2013, pp. 183–197.

[33] D. Lin, R. Raghu, V. Ramamurthy, J. Yu, R. Radhakrishnan, and J. Fernandez, "Unveiling clusters of events for alert and incident management in large-scale enterprise it," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. ACM, 2014, pp. 1630–1639.

[34] J. Xu, Y. Wang, P. Chen, and P. Wang, "Lightweight and adaptive service API performance monitoring in highly dynamic cloud environment," in *2017 IEEE International Conference on Services Computing, SCC 2017, Honolulu, HI, USA, June 25-30, 2017*. IEEE Computer Society, 2017, pp. 35–43.

[35] L. Li, X. Zhang, X. Zhao, H. Zhang, Y. Kang, P. Zhao, B. Qiao, S. He, P. Lee, J. Sun, F. Gao, L. Yang, Q. Lin, S. Rajmohan, Z. Xu, and D. Zhang, "Fighting the fog of war: Automated incident detection for cloud systems," in *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*. USENIX Association, 2021, pp. 131–146.