Journal of Network and Computer Applications 224 (2024) 103842

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Research paper

A survey on intelligent management of alerts and incidents in IT services

Qingyang Yu [a], Nengwen Zhao [a], Mingjie Li [a], Zeyan Li [a], Honglin Wang [b], Wenchi Zhang [b], Kaixin Sui [b], Dan Pei [a,*]

[a] Department of Computer Science and Technology, Tsinghua University, Beijing, China
[b] Bizseer Technology, Beijing, China

ARTICLE INFO

Keywords:
Service system
Alert management
Incident management
ITSM

ABSTRACT

Modern service systems are constantly improving with the development of various IT technologies, leading to a boost in system scales and complex dependencies among service components. The large scale and complexity of services make them more prone to failure. To maintain services' normal and stable operation, alert and incident management (AIM), which analyzes and handles service failures in time, has become an important content of IT service management (ITSM). Many intelligent solutions have been proposed to improve the management process. However, there is currently no comprehensive survey that systematically reviews related works. Moreover, no integrated AIM architecture can cover each detailed process or most existing piecemeal solutions. Therefore, we conduct an in-depth survey to address these problems. To the best of our knowledge, the paper is the most comprehensive survey on intelligent AIM in IT services. Through this survey, we make the following contributions. First, we summarize an integrated architecture that includes detailed AIM processes and key techniques. Second, we provide a systematic review of related works based on the architecture. Third, we give a valuable analysis of current challenges and trends in AIM.

1. Introduction

With the progress of Information Technology (IT), IT services have become indispensable in our daily activities, such as online shopping, social networking, and entertainment. Service providers must maintain the normal and stable operation of IT services to ensure the quality of service (QoS) (Duan et al., 2003). IT service management (ITSM) (Krishnan and Ravindran, 2017) plays a vital role in the daily operations of IT services, including a set of activities to ensure service level agreements (SLAs) (Raimondi et al., 2008).

With the development of virtualization, cloud, big data, and microservice, IT services are becoming more large-scale and complex. Fig. 1 shows an IT service system with numerous components and complex logical relationships (Zhao et al., 2020c). Accordingly, incidents (unplanned interruptions/outages) of the service (Lou et al., 2013) are usually inevitable. Meanwhile, they could destroy service quality and incur huge economic losses. For example, Amazon's one-hour downtime on Prime Day may lead to the loss of up to $100 million in sales.[1] Therefore, incident management (IcM) (Zhou et al., 2017; Chen et al., 2020b) is vital and has become one of the most critical processes in ITSM.

At present, many IT service providers have established systems of alert and incident management (AIM), as shown in Fig. 2. When service users access an IT service, the monitoring system collects data (e.g., metrics (Eyerman and Eeckhout, 2008) and logs (He et al., 2016))

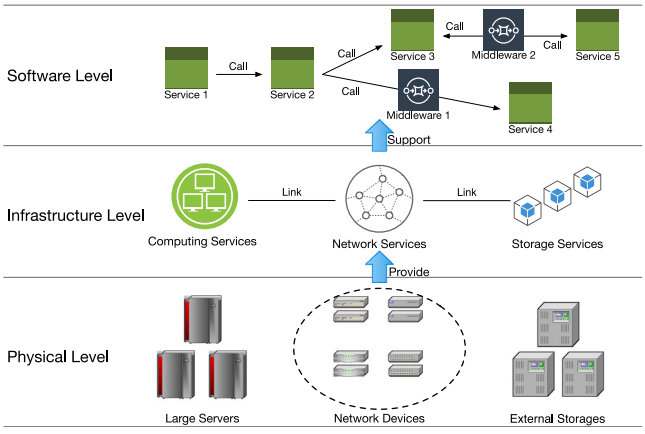


Fig. 1. An example of IT service system.

* Correspondence to: East Main Building Room 9-319, Beijing, Tsinghua University, Haidian District, 100084, China.
E-mail address: peidan@tsinghua.edu.cn (D. Pei).

[1] https://www.businessinsider.com/amazon-prime-day-website-issues-cost-it-millions-in-lost-sales-2018-7.

https://doi.org/10.1016/j.jnca.2024.103842
Received 17 May 2022; Received in revised form 27 August 2022; Accepted 29 January 2024
Available online 20 February 2024

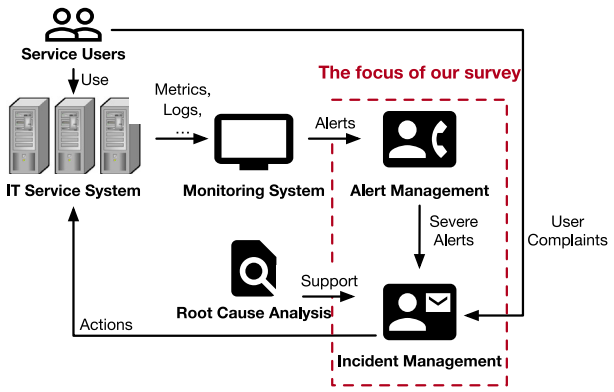


Fig. 2. Overview of AIM in IT service system.

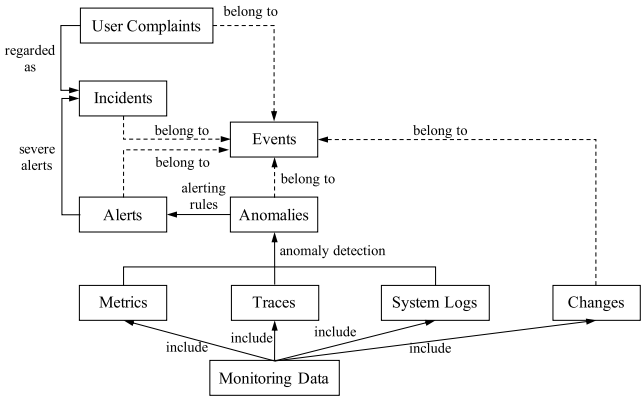


Fig. 3. Relationship of some essential concepts.

from the service system. Once these monitoring data violate predefined alerting rules, alerts would be generated to notify engineers to pay attention to them (Zhao et al., 2020c). Next, alert management is responsible for processing numerous raw alerts from the monitoring system and identifying *severe alerts*. These severe alerts, as well as *user complaints*, are treated as *incidents*, which are further handled by the incident management process. The incident management goals are to assign the incidents to responsible teams, localize the root causes (Solé et al., 2017), and resolve the incidents as soon as possible to ensure service stability.

Traditionally, manual AIM is time-consuming and labor-intensive. In recent years, various solutions (see Section 3.4) have been proposed to handle alerts and incidents in an automatic and intelligent way. Such a body of works also contributes to the operations paradigm shift known as Artificial intelligence for IT Operations (AIOps) (Dang et al., 2019; Notaro et al., 2020; Sabharwal and Bhardwaj, 2022; Rijal et al., 2022), i.e., utilizing Artificial Intelligence to enhance IT Operations.

This study aims to understand AIM by providing a uniform architecture as well as the latest progress in this area. However, to our knowledge, no uniform AIM architecture can fully describe each AIM process or comprise most existing piecemeal solutions. In addition, no thorough AIM survey reviews the state-of-the-art works (see Section 3.3). Therefore, we conducted the survey focusing on AIM (as shown in the red dashed frame of Fig. 2) to tackle these problems. The paper surveys 89 representative research papers from the last 15 years (2008–2022) in detail and makes the main contributions as follows.

1. The paper summarizes an integrated AIM architecture that consists of various processes and can cover most existing piecemeal solutions. This architecture can help readers have a thorough understanding of the AIM workflow and design and evolve the management system for industrial purposes.
2. The paper reviews primary research works on AIM in IT services, following different processes of the summarized architecture. This will help readers quickly understand related works in each specific process they care about to better conduct scientific research and industrial works.
3. The paper provides an in-depth analysis of current challenges and trends in AIM, which points out potential research directions. This can guide the readers to understand the current situation and choose the appropriate direction for research or industrial works.

2. Background

ITSM, which contains a series of management processes and technologies, aims to ensure service quality and user experience (Diao et al., 2016). Besides, Information Technology Infrastructure Library (ITIL) (Arraj, 2010; Potgieter et al., 2005), as a set of detailed and widely accepted best practices of ITSM, focuses on aligning IT services with business requirements. To meet the goals of ITSM/ITIL, the management of alerts and incidents has become the critical content of ITSM/ITIL. For a better understanding of the related background, we will introduce some important concepts (as shown in Fig. 3). As monitoring data that are collected by the monitoring system are the basis of these concepts, we first introduce the monitoring system, which is not our focus but essential to the overall workflow (as shown in Fig. 2). Next, we introduce certain important concepts in ITSM. Finally, we briefly introduce some works on root cause analysis, which can provide support for incident management.

2.1. Monitoring system

The monitoring system (Aceto et al., 2013) is an automatic reactive system that provides an effective and reliable way to ensure service quality. Some well-known organizations provide monitoring services for their products. For example, IBM provides monitoring for their cloud products.[2] Furthermore, many mature monitoring tools are available at present, such as commercial types (e.g., SolarWinds[3]) and open-source types (e.g., Zabbix[4]). A typical monitoring system usually includes certain functions, i.e., data collection and alert generation.

2.1.1. Data collection

Data collected by the monitoring system are the basis for discovering service failures and ensuring the normal and stable operation of the service system. Therefore, it is necessary to continuously and stably collect various system data. These *monitoring data* are the basic data in ITSM, mainly including *metrics*, *traces*, *system logs*, and *changes*.

Metrics. Metrics are essential monitoring data that can indicate the performance of various hardware and software, such as CPU utilization, disk I/O, and service response time. Metrics are one of the most critical monitoring data and are widely used to discover system problems because they are convenient to be visualized and can directly reflect the components' performance status.

System Logs. Most systems record internal operations, status, and errors by logs. Some log generation libraries, such as log4j,[5] and Apache common logging[6] can generate log messages in standard formats, regarded as structural or semi-structural data rather than pure textual data.

[2] https://www.ibm.com/cloud/learn/it-service-management.
[3] https://www.solarwinds.com.
[4] https://www.zabbix.com.
[5] https://logging.apache.org/log4j/2.x.
[6] http://commons.apache.org/proper/commons-logging.

Traces. Traces record the invocation and execution information of service components when executing internal and external requests. There have been some popular tracing technologies to record traces automatically, such as *X-trace* (Fonseca et al., 2007), *Dapper* (Sigelman et al., 2010), *PreciseTracer* (Sang et al., 2011), and *Canopy* (Kaldor et al., 2017). Traces have become important data for monitoring systems, especially since more and more micro-service systems are applied to provide various service functions. They contain rich information related to the call process and the health of requests (e.g., time consumption and return status), indicating both system behavior and performance.

Changes. Change data (Rance, 2011) record each change of the service, which describes the addition, modification, or removal of anything that could have an effect on IT services.

2.1.2. Alert generation

An *anomaly* indicates a state that is inconsistent with the normal behavior of the system, which may cause a system failure. Anomaly detection (Chandola et al., 2009; Hodge and Austin, 2004) is the basis of alert generation, aiming to detect anomalies based on monitoring data using various algorithms or rules. In related literature, a lot of efforts have been devoted to metric anomaly detection (Ren et al., 2019; Laptev et al., 2015; Malhotra et al., 2015), log anomaly detection (Du et al., 2017; Zhang et al., 2019; Yang et al., 2021), and trace anomaly detection (Malhotra et al., 2015; Gan et al., 2019; Nedelkoski et al., 2019). There have been many surveys on anomaly detection from different perspectives, such as network anomaly detection (Ahmed et al., 2016), discrete sequences anomaly detection (Chandola et al., 2010), one class classification (Khan and Madden, 2009), deep learning methods (Chalapathy and Chawla, 2019), and unsupervised outlier detection in high-dimensional numerical data (Zimek et al., 2012). After finding anomalies, the alerts are generated based on some rules. For example, some alerting strategies based on multivariate analysis are introduced in Yang and Guo (2017).

2.2. Related concepts

To better explain the related concepts, we show their relationships in Fig. 3. In addition to the concepts already introduced (i.e., monitoring data, metrics, traces, system logs, changes, and anomalies), there are still some important concepts, namely *alert*, *incident*, and *event* (Brewster et al., 2012; Derdack, 2017; Harper and Tee, 2018).

An *event* can be defined as any occurrence that indicates a change to the current behavior or status of IT services and that has significance for ITSM.

An *alert* is the occurrence of one or a series of events that meet certain criteria. Alerts are usually generated by the monitoring system based on detected anomalies and some alerting rules, e.g., a metric exceeds a predefined threshold for at least a predefined duration. We provide an alert example with several significant attributes, as shown in Table 1. "Alert key" identifies the alert indicator. "Severity" represents the severity level of this alert, which is decided by manual rules. It can only be a reference for determining whether the alert is severe, as the rules cannot always be completely accurate. "Description" specifies the detailed information of the alert, and "Alert ACK" records the alert determination process and the result of the alert, which is usually written by On-Call Engineers (OCEs) or automatically generated by the system. Although "alert" may be called "alarm" (Fournier-Viger et al., 2020) in some papers, this paper uses "alert" instead of it for a unified description.

An *incident* is an unplanned interruption/outage of IT service or degradation of IT service quality. In the IcM system, a reported incident is recorded as a ticket. Incident tickets usually have two primary sources: severe alerts and user complaints. A ticket from a severe alert has similar attributes to this alert, except the alert acknowledgment field is replaced with the resolution field (textual description of incident resolution). In addition, tickets from user complaints are the typical way how the system users express their requests related to incidents in IT services (Jan et al., 2013). Table 2 shows a manual ticket example with several significant attributes. Among them, "Resolution" records the step-wise resolution that describes how this ticketed incident is resolved and may involve multiple expert groups. In the paper, a ticket from an alert is called a *monitoring (incident) ticket*, and a ticket from a user complaint is called a *manual (incident) ticket*,[7].

Table 1
An alert example.

Creation time	Alert key	Severity
2021-3-20 15:29:14	Linux-CPU-CPUUtil	2
Close time	App	Device
2021-3-20 15:37:28	ID(XXX)	IP(*.*.*.*)
Description		
The CPU utilization exceeds the threshold 85% beyond 5 min (current value is 90%).		
Alert ACK		
Contact the responsible expert of App XXX and get a reply that there is no effect on business, then close the alert.		

Table 2
A manual ticket example.

Creation time	Close time	Component
2021-3-22 10:14:24	2021-3-22 11:05:30	Service XXX
Description		
Function A does not respond.		
Resolution		
Step 1. Expert A (network group): Check the network and find it is normal. Step 2. Expert B (database group): Find the database response timed out and solve this issue by increasing memory.		

We further summarize the relationships and differences of the three essential concepts (i.e., alert, incident, and event) as follows. Alerts are a subset of events. As a result, the number of events is larger than that of alerts. Severe alerts and user complaints are two primary components of incidents. As shown in Fig. 2, an alert does not necessarily lead to an incident. Meanwhile, an incident does not necessarily have a reported alert, e.g., some incidents are reported by user complaints only. In terms of severity, incidents are usually more severe than alerts, and alerts are more severe than events in general. Besides, events, alerts, and incidents can all generate notifications to engineers, depending on the specific settings of the organization.

2.3. Root cause analysis

To completely resolve an incident, experts will analyze the root cause of it. Traditionally, experts manually investigate root causes based on their domain knowledge (e.g., the dependencies between service components) and related data analysis (e.g., metrics and logs). However, the manual analysis will cost much time and effort, and is not conducive to the quick resolution of incidents. Therefore, many techniques have been proposed to improve RCA (Łgorzata Steinder and Sethi, 2004; Solé et al., 2017; Soldani and Brogi, 2022). Typically, RCA methods localize root-cause components on a certain level (e.g., services, microservices, containers, servers, or resources). These methods first construct a dependency graph representing the relationship among components, and then localize root-cause components based on the dependency graph.

[7] In this work, we may make exchangeable use of *incident ticket*, and *incident ticket*.

2.3.1. Dependency graph construction

There are two major ways to construct the dependency graph for root cause analysis: system architecture analysis and causality discovery.

The system architecture provides accurate dependencies among components. For example, call relationships of services or microservices can be regarded as dependencies (Kim et al., 2013; Li et al., 2022d,a; Zhang et al., 2021). In this context, if service *A* calls service *B*, then *A* depends on *B*. Thus, the dependency graph can be represented by the call graph generated based on sensors deployed on the components. Resource-sharing relationships among components are also dependencies, which can further enhance the dependency graph (Weng et al., 2018; Wu et al., 2020). For example, if virtual machine *A* and virtual machine *B* are located at the same physical machine, or service *A* and *B* are dependent on the same storage, then they have dependency even if they do not directly call each other. Besides the coarse-grained service-level and virtual machine-level dependencies, traces are utilized in many approaches (e.g., TraceAnomaly (Liu et al., 2020a), MicroRank (Yu et al., 2021), Sage (Gan et al., 2021), TraceRCA (Li et al., 2021b)) to acquire span-level dependencies.

Some other works learn the dependency relationship automatically from the historical monitoring data of the components. The PC algorithm (Spirtes et al., 2000; Kalisch and Bühlman, 2007), a causal relationship learning algorithm, is widely used to identify dependencies in different levels, such as performance metric level (Chen et al., 2014; Ma et al., 2019, 2020; Meng et al., 2020; Lu et al., 2022), API level (Wang et al., 2018a), alert level (Zhang et al., 2020), and microservice level (Lin et al., 2018a). Moreover, the Granger Causality tests (Granger, 1969) are used to infer component dependencies (Thalheim et al., 2017).

2.3.2. Root cause inference

After constructing the dependency graph, a method is in need to infer the root cause based on the dependency graph. Some works (Kim et al., 2013; Weng et al., 2018; Ma et al., 2019; Wang et al., 2018a; Ma et al., 2020; Wu et al., 2020) assume that the metrics of root-cause components are similar to those of the affected components. Thus, a random walker aiming to find the root cause should iteratively go to the neighbors with higher metric similarity. Therefore, they use random walk (Spitzer, 2013) with metric similarity as the transition probability to infer the root cause. Moreover, some works (Chen et al., 2014; Lin et al., 2018a; Liu et al., 2020a, 2021) directly apply a Depth First Search (DFS) method to find the most upstream abnormal component on the dependency graph. In addition, some other methods are used to infer root causes, such as the weighted PageRank algorithm (Liu et al., 2020b; Lu et al., 2022), the conditioned graph traversing algorithm based on the Pearson correlation coefficient (Lin et al., 2018a), the influence maximization algorithm (Zhang et al., 2020), association rule mining (Li et al., 2021b), the counterfactual inference method (Gan et al., 2021), supervised methods (Li et al., 2022d), and other customized RCA methods (Thalheim et al., 2017; Zhang et al., 2021).

3. Overview

At present, there is no comprehensive survey that provides a systematic introduction to the complete relevant works. Moreover, no integrated AIM architecture currently can describe each process in detail or cover most existing piecemeal solutions. Therefore, we conduct an in-depth survey in the AIM field. In this section, we first introduce the survey methodology. Then, based on our survey, we summarize an architecture to describe the complete AIM workflow, which combines critical steps/techniques to help understand each process and potential connections. Next, we analyze related surveys to clarify some contributions of this survey. Finally, on the basis of the architecture, we introduce some detailed statistical information about relevant literature.

Fig. 4. Word cloud of main publishing venues.

3.1. Methodology

We collect literature in a snowballing manner (Wohlin, 2014). We first identify 16 seed papers from those published in relevant conferences and journals in recent years, including SIGKDD '17, ICSE '19–22, ESEC/FSE '20, ASE '19–21, and INFOCOM '20. Then, a keywords-based database search (MacDonell et al., 2010) is performed to make up for the lack of initial literature, using keywords ("alarm", "alert", "incident", and "ticket") to search popular online digital libraries (IEEE Xplore, ACM Digital Library, Springer Online, and Elsevier Online). For the paper screening, we comprehensively consider their quality and relevance to the theme. We notice that many studies are for network security alerts, which analyze network attacks. As monitoring systems for IT services typically generate two types of alerts, i.e., system alerts (as shown in Table 1) and network alerts indicating the network performance and reliability, those network security alert-related works are out of the topic of this survey. After the search, we extend the list of papers by recursively examining those referred by or referring to any included one, which also follows the paper screening above. Finally, we select 89 pieces of literature published from 2008 to 2022.

From the result, we find acquiring related works on AIM challenging as it is an interdisciplinary field that requires familiarity with multiple research areas. The relevant literature is scattered in various publishing venues in different fields, such as Data Mining (e.g., SIGKDD, ICDM, CIKM, DKE), Software Engineering (e.g., ICSE, ESEC/FSE, SCC, ASE), Computer Networks (e.g., SIGCOMM, INFOCOM, NSDI, CNSM, TNSM, CN), Network and Information Security (e.g., NDSS, IM, NOMS, RAID), Computer System (e.g., USENIX ATC, FGCS). Therefore, our literature survey is very valuable for the field. Fig. 4 shows the word cloud of the publishing venues of the literature about AIM, which directly reflects the wide distribution of literature.

3.2. AIM architecture

Based on an extensive survey of relevant literature, we summarize an architecture of AIM (Fig. 5) by classifying existing works and considering the locations and correlations of these classifications. The architecture is an extension of Fig. 2. In the architecture, each rectangle represents a process. This survey focuses on processes within the two red dashed boxes, which represent critical steps/techniques in AIM.

As mentioned earlier, the monitoring system is mainly responsible for two tasks, i.e., data collection and anomaly detection. The data collection process collects various monitoring data that can reflect the system's performance, mainly including metrics, system logs, and traces. The alert generation process triggers alerts based on the anomalies detected from these data and specific alerting rules.

Each alert generated by the monitoring system is sent to the alert management module. Alert management mainly includes three processes: alert correlation, alert storm handling, and alert determination

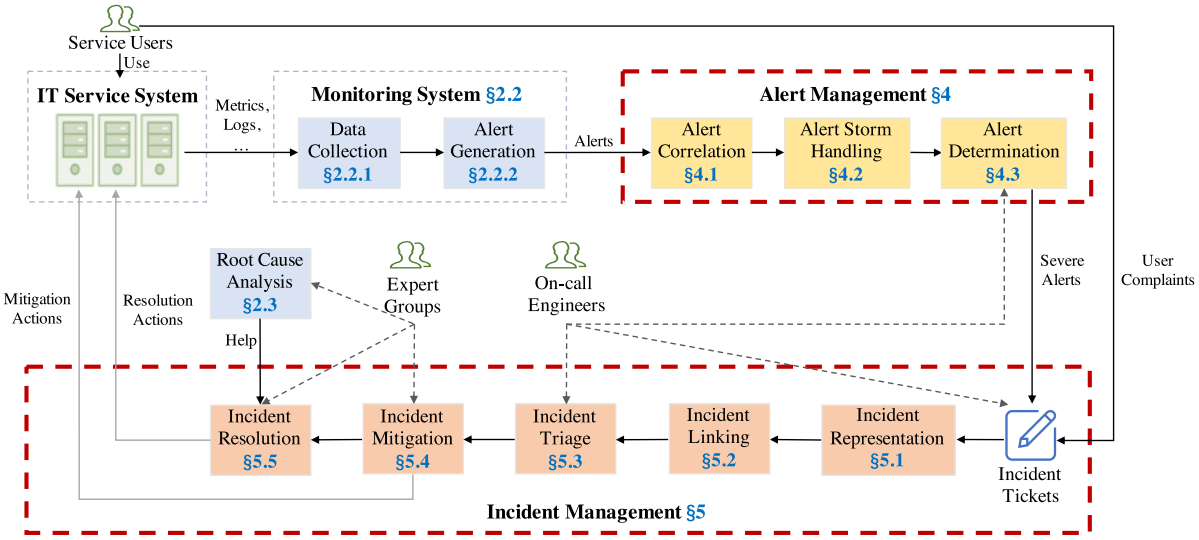


Fig. 5. Illustration of AIM architecture.

(see Section 4). Alert correlation reduces the volume of alerts to be analyzed and provides more valuable information for current problems, as the raw alerts are usually massive and contain miscellaneous information. Alert storm handling resolves extreme alert storm problems when alert correlation techniques are unavailable or do not work well. Alert determination aims to identify severe alerts for creating incident tickets, while non-severe alerts are usually ignored.

The incident management module is responsible for tackling incident tickets and resolving problems described in the tickets (see Section 5). Traditionally, incident management includes three stages, i.e., incident triage, mitigation, and resolution (Chen et al., 2020a). Incident triage is to find the accountable expert (group) for each incident ticket. After an incident is assigned, the corresponding expert first performs mitigation actions quickly to mitigate the incident. Finally, the expert explores the root cause and ultimately acts to resolve the incident based on an in-depth analysis. This process is called incident resolution. This paper first introduces two building blocks for automated ticket analysis, named ticket representation and incident linking. Ticket representation transfers textual tickets into structural data, while incident linking provides a more informative overview of current problems to support the three traditional stages. After that, papers related to the three stages are reviewed respectively.

There are three types of roles in the architecture: service users, OCEs, and experts. If service users encounter issues when using the service, user complaints will be sent to the system and become one of the primary sources of incidents. OCEs, also called system administrators or operators in some papers, are mainly involved in three processes: alert determination, incident ticket creation, and incident triage. Finally, experts, who are also referred to as subject matter experts (SMEs) (Gupta et al., 2008a) or service engineers, perform quick mitigation and final resolution of incidents based on in-depth analysis of the root causes of incidents as well as some prior knowledge and experience.

3.3. Related surveys

To the best of our knowledge, there is currently no comprehensive survey on AIM in IT services, especially no study with the same objective as proposed in this work. The comparison of related surveys is shown in Table 3. Some surveys (Łgorzata Steinder and Sethi, 2004; Soldani and Brogi, 2022) focus on failure RCA, which is only part of our survey's background. Notaro et al. (2021) focus on the AIOps methods for failure management, divided into failure prevention, online failure prediction, failure detection, root cause analysis, and remediation. Only the remediation section covers several works on incident management. Our survey does not cover various fault management works, mainly those based on analyzing alerts and tickets. Some surveys (González-Granadillo et al., 2021; Kotenko et al., 2022) focus on security event management, where the event may go beyond alerts and incidents. And the network security alerts they focus on are not our focus. Some surveys (Mirheidari et al., 2013; Salah et al., 2013) focus on alert correlation, which is only one process of alert management. Paper (Mirheidari et al., 2013) focuses on network security alerts. Paper (Salah et al., 2013) focuses not only on network security alerts but also on network management alerts (Costa et al., 2009), which indicate network performance and reliability and are similar to IT service alerts. Ab Rahman and Choo (2015) survey information security incident handling in the cloud, where the incidents differ from the IT service incidents. Some surveys (Li et al., 2017; Kubiak and Rass, 2018) focus on data-driven techniques in ITSM, which are beyond AIM. Although they cover some incident management papers, they are far less than ours. More specifically, paper (Li et al., 2017) classifies data-driven techniques as multiple tasks, such as log parsing, event generation, classification, clustering, pattern mining, summarization, and problem diagnosis. There are only a dozen works in incident management. Paper (Kubiak and Rass, 2018) surveys ITSM research, mainly including ticket analysis, online failure prediction methods, and IT infrastructure event analysis. The ticket analysis section focuses on ticket classification and only covers about 30 works on incident management.

In contrast, this work will complement existing surveys by providing a uniform AIM architecture and the latest progress in this area. We mainly focus on the AIM works in IT services, where the alerts and incidents are related to service performance and reliability, not security. Critically, this survey summarizes a uniform AIM architecture (Fig. 5) that consists of various processes and can cover most existing piecemeal solutions. And it classifies and introduces related works based on the architecture, which is straightforward to understand. Besides, this survey covers about 30 alert management and 60 incident management research, as shown in Table 4. It is currently the most systematic and comprehensive AIM survey in IT services.

Table 3
Comparison of related surveys.

Category	Survey	Year	Period	Basic topic	Focus domain	Location in this survey
Fault management	Łgorzata Steinder and Sethi (2004)	2004	1988–2002	- Fault localization	- Computer networks	- Background - Alert correlation
	Notaro et al. (2021)	2021	1990–2020	- AIOps methods for failure management	- Computing systems	- Background - Incident management
	Soldani and Brogi (2022)	2022	1992–2021	- Anomaly detection - Failure RCA	- (Micro) Service-based cloud applications	- Background
Event management	González-Granadillo et al. (2021)	2021	2005–2021	- Security event management	- Computer networks	- Alert management
	Kotenko et al. (2022)	2022	2010–2021	- Security event correlation	- Computer networks	- Alert correlation
Alert management	Mirheidari et al. (2013)	2013	2000–2013	- Alert correlation	- Computer networks	- Alert correlation
	Salah et al. (2013)	2013	1988–2011	- Alert correlation	- Telecommunication networks	- Alert correlation - Alert determination
Incident management	Ab Rahman and Choo (2015)	2015	2009–2014	- Information security incident handling	- Cloud environments	- Incident management
IT service management	Li et al. (2017)	2017	1997–2017	- Data-driven techniques in ITSM	- Computing systems	- Incident management
	Kubiak and Rass (2018)	2018	1993–2018	- Data-driven techniques in ITSM	- IT services	- Incident management
Alert and incident management	This	2022	2008–2022	- Alert and Incident management	- IT services	- Alert management - Incident management

3.4. Data source and literature distribution

Due to the privacy of alert and incident data, there are currently only a few open-source datasets. The DARPA1999[8] and DARPA2000[9] are widely-used datasets for network intrusion detection (Man et al., 2012; Alhaj et al., 2016). Landauer et al. (2022) provided a dataset[10] for security alert aggregation research. For the IT service system, a dataset[11] is available in the UCI machine learning repository. It records ticket resolution routing data of an information technology help desk, spanning from March 2016 to February 2017. But the textual descriptions of tickets are not provided. Therefore, this dataset may only be used for ticket reassignment research (Schad et al., 2022). Besides, an alert dataset[12] from a large commercial bank is available, where all sensitive information is anonymized. It includes about 500,000 alert records and alert pattern templates from experts. Thus this dataset can be used for research, such as alert classification and correlation (Chen et al., 2022b). Except for these, almost all alert and incident datasets of IT service systems are private and unpublished.

For readers to better understand the real management of alerts and incidents in different organizations, we classify the primary relevant papers based on data sources divided into IBM, Microsoft, and others. Moreover, Table 4 classifies each work based on its position in the AIM architecture shown in Fig. 5. As a fundamental step, incident representation is common in all incident management processes. Hence, Table 4 does not list incident representation alone. A paper with novel incident representation techniques is classified based on the primary process it focuses on, i.e., incident linking, incident triage, incident mitigation, incident resolution, and other incident analysis tasks. Fig. 6 further presents the number of analyzed works by year of publication.

[8] https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset.

[9] https://www.ll.mit.edu/r-d/datasets/2000-darpa-intrusion-detection-scenario-specific-datasets.

[10] https://github.com/ait-aecid/aecid-alert-aggregation.

[11] https://archive.ics.uci.edu/ml/datasets/Incident+management+process+enriched+event+log#.

[12] https://zenodo.org/record/5336985.

4. Alert management

As shown in Fig. 5, alert management is responsible for identifying severe alerts from raw alerts generated by the monitoring system for creating incident tickets. However, the raw alerts are challenging to analyze due to large numbers, false alerts, etc. In order to solve the problem of alerts' large volume, some works correlate raw alerts to reduce the effort to analyze alerts and provide concise information to describe current alerting problems. We categorize these works as the content of *alert correlation*. Nevertheless, as many service environments do not apply alert correlation technologies or the technologies do not always work, extreme situations called alert storms sometimes occur. So some works focus on handling alert storms. We categorize these works as the content of *alert storm handling*. Last but not least, it is necessary to determine whether alerts are severe and need to be resolved. Traditionally, these tasks are done manually by OCEs, which is inefficient. Therefore, some automated methods are proposed. We categorize these works as the content of *alert determination*.

4.1. Alert correlation

The number of raw alerts is usually huge, but each alert contains only a little information that cannot provide an overview of the current problems. To tackle this problem, alert correlation techniques are proposed to alleviate alert analysis pressure and provide more practical information to describe current issues.

In the intrusion detection system (IDS), two types of works can effectively reduce the effort of analyzing alerts by integrating alerts, i.e., alert aggregation and alert correlation. In this context, alert aggregation identifies and clusters different alerts belonging to a specific attack instance. Alert correlation is to link the alerts from the same attack process together to rebuild the attack course (Chengpo et al., 2006). However, current alert indicators in IT services are much more than the previous intrusion attacks, and there is no clear boundary between alert aggregation and alert correlation as their goals are the same. Moreover, similar works are called alert fusion (Zang et al., 2008) in some literature. For a unified description and understanding, this paper considers alert aggregation and fusion techniques as the content

Table 4
Classification of AIM references.

Reference classification			Data source		
			IBM	Microsoft	Others
Alert management Section 4	Alert correlation Section 4.1		Xu et al. (2017)	–	Costa et al. (2009), Man et al. (2012), Ramaki et al. (2015), Alhaj et al. (2016), Fournier-Viger et al. (2020) and Chen et al. (2022b)
	Alert storm handling Section 4.2		–	Li et al. (2022c)	Yang et al. (2011), Ahmed et al. (2013), Charbonnier et al. (2016), Guo et al. (2017), Lai et al. (2017), Hu et al. (2018), Lin et al. (2018b), Niyazmand and Izadi (2019), Xu et al. (2019), Zhao et al. (2020a) and Landauer et al. (2022)
	Alert determination Section 4.3		Tang et al. (2013c)	Chen et al. (2019c) and Li et al. (2021a)	Jiang et al. (2011), Zong et al. (2014), Hassan et al. (2019), Asres et al. (2020), Zhao et al. (2020c,b) and Li et al. (2022b)
Incident management Section 5	Incident linking Section 5.2		Gupta et al. (2008a,b, 2009), Jan et al. (2013), Mani et al. (2014), Maksai et al. (2014) and Marcu et al. (2009)	Chen et al. (2020b) and Gu et al. (2020)	Lin et al. (2014), Silva et al. (2018), Xu et al. (2020) and Chen et al. (2021)
	Incident triage Section 5.3	Automated triage method Section 5.3.1	Shao et al. (2008a,b), Khan et al. (2009), Miao et al. (2010), Sun et al. (2010), Agarwal et al. (2012), Miao et al. (2012), Dasgupta et al. (2014), Sun et al. (2014), Zeng et al. (2014), Botezatu et al. (2015) and Zeng et al. (2017)	Chen et al. (2019a,b) and Wang et al. (2021)	Motahari-Nezhad and Bartolini (2011), Palshikar et al. (2011), Xu and He (2018), Xu et al. (2018a) and Han and Sun (2020)
		Auxiliary method Section 5.3.2	Diao et al. (2009), Bogojeska et al. (2013, 2014)	Potharaju et al. (2013)	Shimpi et al. (2014) and Xu et al. (2018b)
	Incident mitigation Section 5.4		Wang et al. (2018b)	Jiang et al. (2020) and Wu et al. (2012)	Deb et al. (2017)
	Incident resolution Section 5.5		Tang et al. (2013a), Zhou et al. (2015a,b, 2016), Aggarwal et al. (2016), Wang et al. (2017) and Zhou et al. (2017)	–	Kang et al. (2010)
	Other works Section 5.6		Branch et al. (2014), Giurgiu et al. (2014) and Xu et al. (2016)	Lou et al. (2013, 2017), Chen et al. (2020a,c), Shetty et al. (2021) and Shetty et al. (2022)	–

of alert correlation since they all correlate related alerts together based on certain aspects. Alert correlation has been widely surveyed in the IDS-related fields (Sadoddin and Ghorbani, 2006; Mirheidari et al., 2013; Kotenko et al., 2022) and some other fields (Salah et al., 2013). We introduce some typical alert correlation works that may fit IT service alerts to help understand the field from the perspective of correlation features that they used.

4.1.1. Attribute-based methods

Attribute-based methods correlate alerts based on the similarity of alert attributes. Alerts usually have various attributes (e.g., Table 1). They may be very different depending on their environments (e.g., IT services, networks, industrial systems) and their customized alert formats. Some attributes may be irrelevant or redundant and even have adverse effects. Filtering out these attributes can improve

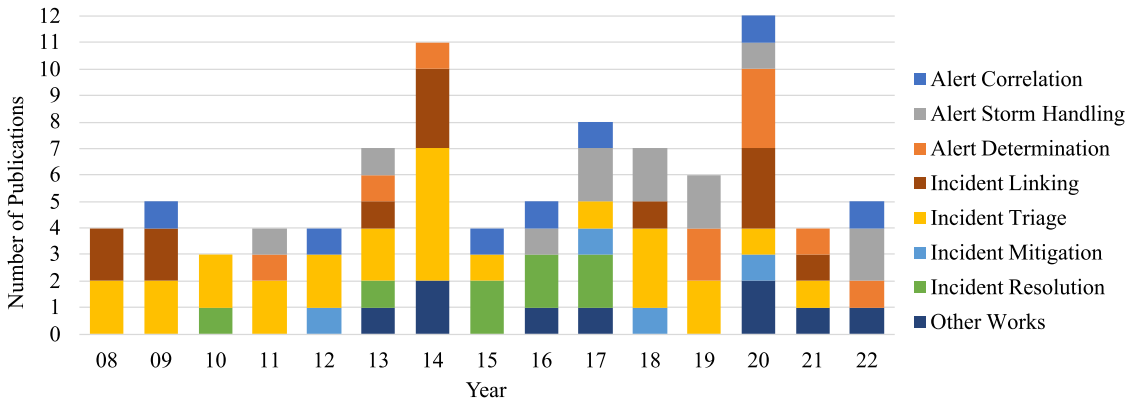


Fig. 6. Number of publications related to AIM surveyed in this paper by year of publication.

computational efficiency and analysis accuracy. Therefore, feature selection is often one basic and important step in many alert analysis works. Traditional feature selection is usually made manually based on knowledge and experience, which may not be accurate enough. A further approach is to use feature engineering (Kuhn and Johnson, 2019) for this task (Asres et al., 2020). Moreover, Alhaj et al. (2016) proposed a 2-tier feature selection method for improving attribute-based alert correlation, which automatically selects appropriate and significant features using Information Gain.

Attribute-based methods are most commonly used for the network (security) alert correlation because alerts with the same source IP address, target IP address, port, etc., are significantly related. For example, a probabilistic strategy (Valdes and Skinner, 2001) is presented to correlate alerts based on the similarity of their overlapping attributes without considering their unique attributes. Besides, Man et al. (2012) proposed a clustering algorithm that iteratively calculates the average value of classes as the new clustering center based on random selection, merging, and dividing dynamically.

For the correlation of service alerts, Lin et al. (2014) proposed a hierarchical clustering method to correlate alerts based on the text. Specifically, it first represents each alert as a bag of words and creates a distance matrix based on natural language processing (NLP) techniques, and uses the Jaccard distance metric to compare any two alerts. Then, alerts are clustered based on the distance matrix using a graph-theoretic approach which uses connected component detection to generate initial clusters before applying the graph-cut algorithm to further refine the clusters. But a purely attribute-based method tends to be less effective because correlating the alerts of the same service, device, etc., is too rudimentary. Besides, correlating alerts based on other attributes, such as the alert key, lacks reasonable explanations, as the alerts of similar indicators in different services or components are not necessarily related. Therefore, few studies correlate service alerts only based on the similarity of service alert attributes.

4.1.2. Dependency-based methods

Dependency-based methods mainly use various known or learned dependencies among alerts to correlate them. Early solutions use predefined static rules to correlate alerts, where the rules are mainly gained from experience and knowledge. Further, some works (e.g., Mannila and Toivonen (1996)) try to mine frequent sequence patterns from alert sequences, which can be used as correlation rules to correlate alerts. But these rules are pre-learned or predefined and cannot adapt to changes in the environment, which are not intelligent enough. Therefore, Costa et al. (2009) proposed a smart and adaptive alert correlation architecture, which uses a rule discovery approach to generate the rules based on alerts and incident tickets. In particular, the approach is independent of the network topology and uses incident ticket information to get feedback from the correlation results based on an association rule algorithm.

However, in modern large-scale service systems, service components are numerous and have complex dependencies. A failure of one component may cause a series of alerts. The huge alert number challenges sequence mining methods. Therefore, some works correlate alerts based on the dependencies between components or indicators instead of alert sequences. The core of these works is to find the dependencies. Thus some dependency learning methods in Section 2.3.1 can be used to achieve this goal. Then the alerts can be correlated based on the dependencies. Besides, some statistical approaches are adopted to learn the dependencies of alerts, such as Pearson correlation coefficient (Mahimkar et al., 2009) and Granger causality (Yin and Yao, 2016). But Pearson correlation coefficient cannot discover the non-linear or non-functional relationships amongst alerts, and Granger causality assumes the lags between two alerts are stable, which does not always hold in IT service alerts. Considering the Maximal Information Coefficient (MIC) (Reshef et al., 2011) is adequate to measure the strength of the association between two variables, Xu et al. (2017) proposed a MIC-based correlation algorithm to group alerts into a higher-level and more-informative aggregated alert. In addition, Fournier-Viger et al. (2020) used the network topology to find dependencies among network alerts. Specifically, they modeled a real-life telecommunication network as a dynamic attributed graph and proposed a correlation measure to extract correlation patterns. Ramaki et al. (2015) used Bayesian networks to construct the dependencies between alerts and correlated alerts.

4.1.3. Hybrid methods

Hybrid methods attempt to correlate alerts using both their attributes and their dependencies. Deep learning provides the feasibility of integrating the semantic information contained in alert attributes and the dependencies between alerts. Chen et al. (2022b) proposed a supervised framework, OAS (Online Alert Summarizing), to automatically correlate alerts online. OAS contains four main components, ASR (Alert Semantics Representation), ABR (Alert Behavior Representation), ACT (Alert CorrelaTion), and online summarizing. OAS first respectively learns two types of alert information, i.e., semantic information and behavior information, based on two deep learning models, i.e., ASR and ABR. ASR integrates the contextual information of alert words according to their importance. ABR mines the common behavior pattern between alerts from the alert occurrence series. Then, OAS adopts the deep learning model ACT to combine the above two information and determine the correlation between alerts. Online summarizing adopts these trained models to summarize the newly reported alert online by a time window. More specifically, in the online stage, for a new alert and any previous alert in a time window, OAS can represent their semantic information and behavior information by ASR and ABR and obtain the correlation degree between the two alerts based on ACT. Thus the most correlated alert to the new one can be found. Then, add the new one into the incident (set) of its most correlated alert. Otherwise, form a

new set for the new alert. Such a strategy avoids modifying previous alerts and ensures that each alert is processed only once in the online summarizing stage.

4.1.4. Summary

Alert correlation techniques mainly have two application scenarios. One is correlating real-time alerts to provide intuitive and effective information reflecting current problems. Another is correlating current alerts with similar historical alerts for using historical handing information to analyze current issues. As mentioned earlier, attribute-based methods combine multiple attributes to correlate alerts, which can uncover associations that are challenging to find with dependent-based methods. These methods can work for both two scenarios. However, the interpretability of these methods is lacking. In contrast, the dependency-based methods are more interpretable as they correlate alerts using the dependency relationships of alerts or their components and indicators. Moreover, the dependencies can provide engineers with more intuitive and practical information to analyze current problems. But dependency-based methods mainly work for the first scenario as they cannot correlate similar alerts. Hybrid methods combine alert attributes and dependencies. Thus they should be more effective, fitting both two scenarios. But they usually require large and comprehensive labeled data, which are challenging to obtain.

4.2. Alert storm handling

In real-world ITSM, the alert number sometimes far exceeds what engineers can handle. Such an extreme situation is called the *alert storm*. As a real and ubiquitous phenomenon in the IT service environment, the alert storm has become one major pain point in the current alert management.

Why do alert storms occur? The alert storm occurs mainly due to the large scale and complexity of the service system (e.g., Fig. 1). On the one hand, the failures of common components used by multiple services cause these services to be affected and many alerts generated. On the other hand, the propagation of failures between related components further exacerbates the generation of alerts. Intuitively, the alert correlation can effectively handle alert storms by correlating relevant alerts to reduce the analysis effort. However, since the alert correlation methods are not used in some service environments, or the actual effectiveness of these methods is not ideal, alert storms still occur.

4.2.1. Industrial system alert storm

In industrial systems, a phenomenon similar to the service alert storm is called alert flood (Wang et al., 2015), in which the alerts exceed what operators can handle adequately. Some previous standards, such as ISA 18.2 (International Society of Automation (ISA), 2009) and EEMUA 191 (Equipment and Association, 1999), suggest identifying the alert flood when the alerting rate is higher than ten alerts per 10 min per operator (Ahmed et al., 2013; Lai et al., 2017). Although the alert number of the alert flood is far less than that of the service alert storm (Zhao et al., 2020a), the methods of handling alert floods may be exploited for service alert storms.

Most works on handling alert floods analyze alert sequences to find the typical patterns and get valuable information. Ahmed et al. (2013) proposed a pattern matching algorithm based on dynamic time warping (DTW) to measure the similarity between two alert sequences. Charbonnier et al. (2016) proposed a method to extract fault templates from alert lists recorded during different faults. Alert lists of the same fault are condensed into a weighted sequential fault template to represent the fault with a unique alert sequence. Combining the templates with a weighted similarity measure can propose possible faults for alert floods. Guo et al. (2017) proposed a match-based accelerated alignment (MAA) method to assess the similarity of alert sequences with good robustness. Lai et al. (2017) proposed an algorithm to match an online alert sequence with a pattern database and calculate similarity, incorporating some technologies to improve effectiveness and efficiency. Niyazmand and Izadi (2019) proposed a modified PrefixSpan algorithm to identify and categorize sequence patterns in alert floods. This algorithm was claimed to work more efficiently than clustering methods of pattern recognition. Furthermore, Hu et al. (2018) studied the frequent alert patterns to facilitate the configuration of dynamic alert suppression, which is a commonly used technique to reduce the alerts for analysis by temporally suppressing some irrelevant or unimportant alerts. Xu et al. (2019) proposed a method to predict upcoming alerts for a current alert sequence by exploiting similar historical alert flood sequences based on a Bayesian estimator.

4.2.2. Network alert storm

In a network alert management system in about 2011, Yang et al. (2011) found that memory depletion of the system is not caused by a high alert arrival rate but by excessive system process delays. When the system is required to maintain a state of alerts, it needs to keep some alert information describing ongoing transactions of these alerts for a certain period. In such a case, all memory may be occupied and unable to process new alerts. They explored an approach to managing the network alert storm by calculating inter-process latency, which is the time alerts spend waiting to be handled, based on dynamic states of alerts and behaviors of system performance. It provides QOS trending projection functionality of inter-process latency to manage CPU and memory capacity and to trigger the control mechanism to protect system resources. However, system resources are usually not the bottleneck in the current alert management system. The main challenge is that operators cannot identify real issues that need to be addressed from the alert storm.

For network security alert storms, Landauer et al. (2022) presented a framework for automatic and domain-independent alert aggregation, which does not need specific alert formats. The approach mainly consists of rule-based algorithms that group alerts by occurrence times, cluster these groups by similarity, and extract commonalities to model meta-alerts without merging all considered alerts into a single common format. Based on these metrics and techniques, they proposed an incremental procedure for generating abstract alert patterns that enable continuous classification of incoming alerts. As a result, the approach transforms alerts generated by IDS into higher-level meta-alerts that represent specific attack patterns. Some strategies of this framework can also be applied to IT service alert storms.

4.2.3. Service alert storm

For IT service systems, the alerts have richer content shown in Table 1. Therefore, the alert content deserves to be analyzed instead of only the alert sequence. To better understand and solve the alert storm problem, Zhao et al. (2020a) conducted the first empirical study of the IT service alert storm based on large-scale real-world alert data. The results show that some alerts in the alert storm are irrelevant to the failure, and many alerts relevant to the failure have some correlations, i.e., textual and topological correlations. Based on these findings, they proposed an alert storm detection method to identify the alert storm accurately and an alert storm summary method to recommend a small set of representative alerts to engineers for failure diagnosis. More specifically, they leveraged Extremely Value Theory (EVT) (Siffer et al., 2017) to detect alert storms adaptively and accomplished a three-step alert storm summary method.

In addition, Li et al. (2022c) proposed a strategy for resolving alert storms (called flooding incidents in their paper). Considering the incidents here are homogeneous with alerts, the strategy can also be regarded as handling alert storms. The strategy includes two aspects: (1) provide a global incident view by linking related incidents; (2) prioritize high-impact incidents out of all incidents. More specifically, they correlate incidents using their proposed methods (LiDAR (Chen et al., 2020b) and COT (Wang et al., 2021)). LiDAR leverages both textual information and components inter-dependency information to

calculate the linking probability of two incidents. COT further correlates and clusters a set of related incidents among services. Then, they use their proposed deep learning method DeepIP (Chen et al., 2020a) to prioritize incidents with three inputs: incident description, key terms (such as API names), and runtime environment information (such as incident-occurring device).

4.3. Alert determination

In many IT service systems, especially business-sensitive services (such as digital banking), the monitoring systems report all suspicious alerts that may affect the systems' normal operations to avoid missing potential incidents. This way causes reporting many false alerts or low-priority alerts that cost much effort to handle. Even if some organizations optimize the alerting rules as much as possible, false alerts and missed alerts still abound. Alert determination is to identify real alerts or problems that need to be ticketed for subsequent handling from many alerts. Traditionally, the alert determination is manually done by OCEs. Unfortunately, manual analysis leads to low efficiency. Sometimes the alerts even exceed what OCEs can properly investigate (Hassan et al., 2019). Therefore, some automated methods were proposed to improve the efficiency and effectiveness of the manual way. These methods can be divided into three types: *alert distinguishing*, which determines whether an alert is real; *severe alert identification*, which discovers a few critical alerts from many alerts; and *alert-based incident identification*, which identifies incidents from a group of alerts.

4.3.1. Alert distinguishing

Alert distinguishing methods identify the authenticity of each alert. Tang et al. (2013c) proposed an alert predictor to assign each alert a label (real or false), built on a set of predictive rules automatically generated by a rule-based learning algorithm (Srikant and Agrawal, 1996) based on historical alerts. Hassan et al. (2019) presented an automated method that uses historical and contextual information of alerts to distinguish them. Specifically, it generates alert causal dependency graphs and assigns an anomaly score to each edge based on the historical occurring frequency. Then, it uses a network diffusion algorithm to propagate anomaly scores in dependency graphs and generates aggregate anomaly scores for each alert. At last, it uses these scores to classify alerts as real or false. Li et al. (2022b) proposed a method of combining machine learning and deep learning to detect false alerts. They first performed feature engineering to select alert attributes by manually viewing data content and distribution. They removed the meaningless attributes (e.g., alert id) and the attributes with too many missing values. While training, they used the deep neural network (DNN) hidden layer output features to train the machine learning model. They found that deep learning can improve the classification effect of traditional machine learning models through experiments and recommended using the features of DNN hidden layer output to train traditional machine learning models for false alert detection.

4.3.2. Severe alert identification

Severe alerts refer to the critical alerts that really affect the system's normal operation in a large set of alerts. Therefore, it is crucial to identify severe alerts to effectively reduce the alerts to be analyzed and processed.

Most related works use ranking-based methods. Jiang et al. (2011) proposed a simple alert ranking strategy based on the linear relationships between alert thresholds. Specifically, they used invariant networks to map the local thresholds of various rules to equivalent values in the global context to rank the alerts. However, the assumption of a linear relationship between two KPIs may not always hold in reality. Lin et al. (2018b) proposed CAR, a collaborative alert ranking framework that exploits both temporal and content correlations from heterogeneous categorical alerts. Zhao et al. (2020c) proposed AlertRank, a ranking-based framework for identifying severe alerts automatically and adaptively. Specifically, AlertRank extracts a set of powerful and interpretable features to characterize the severities of alerts. Then, AlertRank adopts the popular XGBoost ranking algorithm (Chen and Guestrin, 2016) to identify the severe alerts from all incoming alerts based on the aforementioned features.

In addition, Zong et al. (2014) proposed a scalable critical alert mining framework to find a set of k critical alerts such that the number of alerts potentially triggered by them is maximized. Based on this idea, a series of methods were proposed or applied to solve this problem and improve the scalability and performance of the framework, such as existing Granger causality analysis tools (Seth, 2010) to mine the dependency rules of alerts and conditional probabilities to estimate the uncertainty of the rules (Kim and Brown, 2010).

4.3.3. Alert-based incident identification

Different from previously mentioned methods of analyzing a single alert to determine its authenticity or to identify its severity, alert-based incident identification directly identifies incidents from a great number of alerts to create tickets.

Asres et al. (2020) proposed a supervised ticket prediction system. The system uses a sliding time window and feature engineering for feature extraction from related history alert streams and uses binary classifiers using a gradient boosted decision tree algorithm for modeling.

Chen et al. (2019c) developed AirAlert to forecast incidents. It works as a global watcher for the entire cloud system, collects all alerts, detects dependency among alerts, and proactively predicts incidents. More specifically, it analyzes the relationships between incidents and alerts leveraging Bayesian network and predicts incidents using XGBoost. Li et al. (2021a) further proposed Warden to automatically detect incidents based on alerts. They train an inference model based on historic failure patterns. Upon detecting potential incidents, Warden extracts a set of representative alerts to notify relevant OCEs. Experiment results show that Warden is more effective than AirAlert.

Zhao et al. (2020b) proposed eWarn, an interpretable approach to predict incidents based on alerts within a particular time horizon. EWarn extracts a set of features to represent omen alert patterns via feature engineering. And it incorporates multi-instance learning (Carbonneau et al., 2018) to reduce the influence of noisy alerts irrelevant to the occurring incidents. Moreover, it utilizes LIME (Ribeiro et al., 2016), an explanation technique, to generate a report for engineers to interpret the prediction result in a visualization manner.

4.3.4. Summary

Alert distinguishing provides an intuitive prediction for the authenticity of each alert so that only the alerts identified as real ones need to be further handled. However, many alerts may be identified as real, affecting the efficiency of subsequent handling. Severe alert identification sorts the severity of alerts to identify severe alerts, which can provide more detailed information to help experts choose which alerts to handle first. But a few individual alerts hardly provides an overview of the current incident. Alert-based incident identification directly extracts incidents from many alerts, which seems most efficient but may ignore some critical alert information.

5. Incident management

As shown in Fig. 5, incident management mainly includes five processes: incident representation, incident linking, incident triage, incident mitigation, and incident resolution. In order to accomplish incident management intelligently, the first is to represent incident tickets in a numeric form that the machine can utilize. So we will first introduce typical ticket text representation methods for automated analysis and then introduce the research works on different IcM processes.

5.1. Incident representation

Representing the incident ticket text as a numeric form that subsequent methods can use is the basis of the automated analysis. Ticket representation typically starts from textual featurization, in which the ticket content is divided into features (Diao et al., 2009). The typical way represents the ticket text as a bag of words divided based on space or symbols. Then, these words are transformed into a vector space model for subsequent analysis. However, the raw text of tickets is challenging to use. Correspondingly, some methods are used to solve existing challenges. The main challenges and solutions are as follows.

- The descriptive text manually recorded may have many variations. First, spelling errors are common and unavoidable. So some spelling correction techniques (Hládek et al., 2020) can be used. Second, a word may have different forms (e.g., different tenses and voices, singular and plural). Hence, some stemming techniques (Moral et al., 2014; Singh and Gupta, 2017) can be applied, such as Porter's algorithm (Porter, 1980), which is one of the most classic stemming techniques. Third, some synonyms and compound terms may affect analysis results. Hence, synonym detection and compound term processing techniques can be used (Shimpi et al., 2014; Yu, 2008).
- The words on tickets have different parts of speech. Building Part-Of-Speech (POS) tags for the words is important for text analysis. To solve this challenge, some NLP methods, such as Stanford POS Tagger (Toutanova et al., 2003; Toutanova and Manning, 2000), are used to identify the nouns, verbs, etc., in the ticket text (Agarwal et al., 2012; Wang et al., 2017).
- Some words are meaningless and even adversely affect the subsequent analysis. These words are regarded as stopwords. Filtering out stopwords is essential. Many works manually define stopwords based on domain knowledge. For example, Shimpi et al. (2014) prepared an exclude list that contains some useless words to remove. Furthermore, some works use automated methods to find stopwords. For instance, Lin et al. (2014) thought the most frequently occurring words in ticket texts were more likely to be meaningless stopwords. They sorted all words in descending order of total count and chose the top k words as stopwords.

In addition, Tang et al. (2013a) proposed a representation method for monitoring tickets based on attribute-based features, including categorical features (e.g., hostname, process name), numeric features (e.g., CPU utilization, disk free space percentage), and textual features (e.g., incident description). Zhou et al. (2015b) and Zhou et al. (2016) found that using semantic features to represent monitoring tickets was better than using attribute-level features. They proposed a feature extraction approach to represent ticket information using topic-level features obtained via the Latent Dirichlet Allocation (LDA) model (Blei et al., 2003). Considering that phrases contain richer and more practical information for problem analysis than words do, Wang et al. (2017) proposed a domain-specific approach for extracting useful phrases (e.g., "available disk space" and "backup client connection") to represent tickets. Specifically, the data compression algorithm Lempel–Ziv–Welch (LZW) (Welch, 1984) is used to extract hot phrases. Then, the Aho–Corasick (AC) algorithm (Aho and Corasick, 1975) is used to locate all occurrences of phrases for finding similar problems in historical tickets to solve the current problem.

Recently, deep learning methods have been widely used in ticket text representation. A study Chen et al. (2019b) suggested that many special terms (e.g., API names and component names) are helpful in conducting incident triage. However, traditional text encoding methods either ignore these special terms or treat them as "unknown words" uniformly (Joulin et al., 2016). Moreover, traditional text encoding methods cannot properly encode special terms due to their low frequency. Therefore, Chen et al. (2019b) adopted a Convolutional Neural Network (CNN) based neural-language model (Johnson and Zhang, 2017) to perform domain-specific text encoding.

In addition, Gu et al. (2020) employed an advanced pre-trained natural language model (BERT (Devlin et al., 2018)) to extract word-level features from manual tickets and used a neural network with one hidden layer to generate the word representations of the monitoring tickets. Chen et al. (2020b) proposed a three-stage textual representation method that includes word-level embedding, deep feature mapping, and semantic comprehension. Chen et al. (2021) proposed a method based on graph representation learning, which learns an embedding vector for different types of incidents. Such representation encodes not only the temporal locality of incidents but also their topological relationship. Furthermore, Deep Structured Semantic Model (DSSM) (Huang et al., 2013) and its variations (e.g., Dai et al. (2018), Gao et al. (2014) and Elkahky et al. (2015)) can generate hidden representations for two inputs and compute the relevance score (e.g., cosine similarity) between them for text matching.

5.2. Incident linking

Incident linking (also called incident correlation or incident aggregation) refers to linking an incident with related incidents or other items. The linking results can provide OCEs and experts with more intuitive information to help them dispatch tickets and analyze incidents. Correspondingly, some methods have been proposed. As monitoring tickets are homogeneous with alerts, thus the methods to correlate alerts (Section 4.1) can be used for linking monitoring tickets. Besides, the methods focusing on incident linking can be divided into three categories (see Table 5).

5.2.1. CMDB-based methods

The configuration management database (CMDB) provides the relations among configuration items (CIs). Some works focus on recognizing the CIs from tickets to further link tickets based on the CMDB. For example, Gupta et al. (2008a,b) proposed a method for linking incidents described in manual tickets with CIs, using information integration techniques and machine learning. The method can further identify failure components by efficiently browsing relationships among CIs. Moreover, they Gupta et al. (2009) presented an approach to link multi-dimensional knowledge (e.g., configuration data, system vital data, log data, and tickets). More specifically, it can achieve automated ticket classification, automated association of resources with tickets based on integration with CMDB, and collection of system vitals relevant to the ticket through integration with monitoring systems. Marcu et al. (2009) presented a step-wise method to correlate manual tickets with monitoring tickets by leveraging insights from service definition and description of the deployed infrastructure in CMDB. In this method, service categorization is augmented with service/resource similarity to facilitate the selection of resources that demonstrate the correlation between tickets. Jan et al. (2013) presented a statistical learning method called Conditional Random Field (CRF) to automatically identify server names in free-text tickets. With server names, various information can be integrated by extending CMDB and linking together all the information pertaining to the server.

5.2.2. Clustering-based methods

These methods use clustering techniques to link incidents. Considering the difference between monitoring tickets and manual tickets, Lin et al. (2014) proposed two distinct frameworks with different graph-theoretic approaches to cluster the two kinds of tickets, respectively. Specifically, the Jaccard distance metric and a top-down clustering approach using connected component detection and graph-cut are applied to cluster semi-structured text in monitoring tickets. As for manual tickets, an approach based on matrix factorization and KD-tree is proposed to cluster unstructured text. Mani et al. (2014) employed a latent semantic indexing based technique to cluster large

Table 5
Summary of references about incident linking.

Category	Reference	Linking item				Main technique
		Monitoring ticket	Manual ticket	CI	Other	
CMDB-based method	Gupta et al. (2008a) and Gupta et al. (2008b)		√	√		Knowledge Engineering & Information retrieval
	Gupta et al. (2009)	√	√	√	√	Knowledge Engineering & Information retrieval
	Marcu et al. (2009)	√	√			Step-wise correlation
	Jan et al. (2013)		√	√		Conditional Random Field
Clustering-based method	Lin et al. (2014)	√	√			Hierarchical clustering
	Mani et al. (2014)	√	√			Clustering & Cluster merging technique
	Maksai et al. (2014)	√				Hierarchical clustering & Active learning
	Silva et al. (2018)	√				Hierarchical clustering
	Chen et al. (2021)	√				Graph representation learning
Supervised learning based method	Xu et al. (2020)	√	√			Multi-view similarity measure
	Chen et al. (2020b)	√	√			Deep learning
	Gu et al. (2020)	√	√			Transfer learning

non-regular unstructured text and a hierarchical n-gram based technique to cluster short snippets of semi-structured text. They further developed a cluster merging technique to merge the results from the two complementary algorithms. Maksai et al. (2014) devised a two-stage technique to classify incident tickets. In the first stage, tickets are grouped using hierarchical clustering, and each cluster is labeled using appropriately chosen sample tickets. In the second stage, the ticket classification for each ticket class is refined by either another round of hierarchical clustering or active learning. Silva et al. (2018) provided a hierarchical clustering approach to identify similar incidents, based on the relationship that similar incidents may share the same set of events. The distance between different incidents is calculated based on the proportion of same events, and then clustering is performed. However, this relationship may not be satisfied in other environments. Chen et al. (2021) proposed GRLIA, an incident aggregation framework based on graph representation learning. GRLIA comprises two incident linking methods. In the offline analysis phase, the first method links the incidents that may be triggered by each individual failure based on a well-known community detection algorithm called Louvain algorithm. Then an embedding vector is learned for different types of incidents by leveraging existing graph representation learning models. In the online incident linking phase, the second method employs the learned incident representation to link incidents by considering their cosine similarity and topological distance.

5.2.3. Supervised learning based methods

These methods use supervised learning methods to link incidents. Xu et al. (2020) proposed a multi-view similarity measure framework to integrate several kinds of existing similarity measures, including surface matching based, semantic-based, and syntax-based measures. A supervised method is used to determine the similarity threshold score for each measure. In particular, they used a machine learning based policy to integrate various similarity measures in a more general way, which makes the framework flexible and extensible. Chen et al. (2020b) proposed a deep learning based approach LiDAR, which incorporates the textual description of incidents and structural information of component dependencies extracted from historically linked incidents. Gu et al. (2020) conducted an empirical study on Microsoft's production cloud service systems and found about 77.70% of the system (monitoring) incidents that affect customers can be detected before any customer reports them. They proposed LinkCM, a transfer learning based approach, to automatically link manual tickets with monitoring ones to improve the triage efficiency.

5.3. Incident triage

Incident triage is to find a suitable expert (group) for an incident ticket to solve it. Traditionally, OCEs manually finish this task, requiring specific domain knowledge to identify where the problem lies, e.g., database, storage, and network. However, it is challenging for OCEs to master various domain knowledge. Furthermore, IT services' large scale and complexity increase the challenge as needing many expert groups responsible for different system components. So the manual way is usually inefficient, costing much time and human effort. Therefore, some automated methods have been proposed to improve efficiency and effectiveness from two perspectives, i.e., automating ticket triage and providing auxiliary information to OCEs to help triage.

5.3.1. Automated triage methods

The goal of automated triage is to automatically find the correct experts for tickets. Ticket triage can also be regarded as one content of expert recommendation, which is also known as expert finding, expert retrieval, or expert search. However, expert recommendation far exceeds the field of ticket triage, and there are many related works (Nikzad-Khasmakhi et al., 2019; Lin et al., 2017; Yuan et al., 2020). Correspondingly, this section introduces only the works on the ticket triage field. We summarize various related works based on the features they use (see Table 6).

Content-based Methods. The content-based methods deal with ticket triage only using the text information of tickets, considering historical tickets contain associations between experts and ticket descriptions. Most of these works Agarwal et al. (2012), Dasgupta et al. (2014) and Zeng et al. (2014) consider ticket triage as a text classification problem. Godbole and Roy (2008) and Agarwal et al. (2012) suggested the Support Vector Machine (SVM) (Joachims, 1998) to be generally superior to some other technologies, such as decision trees, Bayesian networks, and K-means, in this field. Accordingly, SVM became a widely used classifier for ticket triage. Agarwal et al. (2012) used two attributes (i.e., description and resolution group) of historical manual tickets to train a classifier for resolution groups based on the combination of SVM and a discriminative term based approach. Then the classifier can predict the most appropriate resolution group for a new ticket. Dasgupta et al. (2014) presented an approach that comprised a correlation model and a classification model. The former model correlates different data sources (alerts and manual tickets) to obtain a richer text based on multiple parameters and domain information. The latter model classifies the tickets based on the SVM method with a Radial Basis Function (RBF) Kernel.

Furthermore, Zeng et al. (2014) regarded ticket triage as a hierarchical classification problem where an instance can be labeled with nodes belonging to more than one path or a path without ending on a leaf in the hierarchy. They proposed a hierarchical multi-label classification method with a contextual hierarchy loss function that classifies the monitoring tickets using Bayesian decision theory.

Sequence-based Methods. Content-based methods usually focus on one-time assignments for tickets but do not consider continuous routing. However, in practice, tickets may not be directly assigned to the right groups for the first time. An empirical study (Chen et al.,

Table 6
Summary of references about automated ticket triage based on their categories, the applicable tasks, the ticket types, main techniques, and the main features used (C = Content, S = Sequence, EA = Expert Ability, O = Other).

Category	Reference	Applicable task		Ticket type		Main technique	Main feature used			
		Assign	Reassign	Monitor	Manual		C	S	EA	O
Content-based Method	Agarwal et al. (2012)	√			√	SVM	√			
	Dasgupta et al. (2014)	√		√	√	SVM	√			
	Zeng et al. (2014)	√		√		Hierarchical classification	√			
Sequence-based Method	Shao et al. (2008b,a)		√	√	√	Markov model		√		
	Miao et al. (2012)		√	√	√	Generative model		√		
	Sun et al. (2014)		√	√	√	Generative model		√		
Hybrid method	Sun et al. (2010)	√	√		√	Sequence mining	√	√		
	Motahari-Nezhad and Bartolini (2011)		√	√		Information retrieval	√	√		
	Khan et al. (2009)	√	√	√		Generative model	√	√	√	
	Miao et al. (2010)		√		√	Generative model	√	√	√	
	Palshikar et al. (2011)	√		√		Statistics-based algorithm	√		√	
	Botezatu et al. (2015)	√			√	Multi-view clustering	√		√	
	Zeng et al. (2017)	√		√		Hierarchical classification	√			√
	Xu and He (2018)	√	√		√	Generative model	√	√	√	
	Xu et al. (2018a)	√	√	√	√	Generative model	√	√	√	
	Han and Sun (2020)	√	√		√	DNN	√	√		
	Chen et al. (2019b)	√	√	√		GRU	√			√
	Wang et al. (2021)	√		√	√	SVM & decision tree	√			√

2019a) about incident triage conducted on 20 large-scale online service systems in Microsoft showed that incorrect assignments occurred frequently, and many incidents were reassigned at least once.

One-time assignment methods can also be used for reassignments by repeatedly executing them and selecting the next best item. However, the items selected by these methods often have strong similarities due to these methods' mechanisms. In practice, the group that can resolve an incident may be very different from the current group, such as the database group and network group. Therefore, one-time assignment methods are not the best for reassignments.

Due to the service system's large scale and complexity, an incident ticket's dispatching process may involve multiple expert groups and form a routing sequence until it reaches the expert who can handle it. The ticket routing sequence describes a ticket dispatch's complete processes, and each process represents one expert's actions that handle the incident. Considering the fact that historical routing sequences provide rich information about the relationship between experts, some works (Shao et al., 2008b,a; Miao et al., 2012; Sun et al., 2014) complete ticket reassignments utilizing only the routing information. The methods are called sequence-based methods.

Shao et al. (2008b,a) developed EasyTicket, a ticket routing recommendation tool which comprises a Markov model to capture the transfer decisions from historical resolution sequences and a search algorithm to generate transfer recommendations without accessing ticket content. Some other works (Miao et al., 2012; Sun et al., 2014) take routing sequences as collaborative networks, a particular type of social network formed by participants who collectively achieve specific goals, such as fixing software bugs and resolving incidents. Miao et al. (2012) proposed a network model to simulate real collaborative networks and a routing model to simulate human collaboration dynamics. This model can jointly simulate the structure and the ticket routing procedure. In contrast, Sun et al. (2014) directly inferred the routing models in real collaborative networks without simulating a collaborative network. More specifically, they formalized multiple routing patterns into a generative model to formulate experts' routing decisions.

Hybrid Methods. Hybrid methods deal with the triage tasks by utilizing multiple information (e.g., ticket content, routing sequence, and expert ability). Some works Sun et al. (2010) and Motahari-Nezhad and Bartolini (2011) suggested hybrid methods were more effective than methods that use either resolution sequence or ticket content alone. They intended to improve the methods by using both content and sequence information. Specifically, Sun et al. (2010) extended the sequence-only approach in Shao et al. (2008b) by further mining the description information of tickets and proposed a content-aware sequence mining technique to build ticket routing models which can filter

candidate groups in routing based on ticket content similarity. Motahari and Bartolini (Motahari-Nezhad and Bartolini, 2011) regarded incident triage as an information retrieval problem, and they combined information retrieval techniques and a process mining technique to recommend the best next resolution steps and experts.

Some works (Khan et al., 2009; Miao et al., 2010; Palshikar et al., 2011; Botezatu et al., 2015; Xu and He, 2018; Xu et al., 2018a) combine expert ability for better ticket triage. Khan et al. (2009) proposed an automated ticket routing framework, which builds a routing prediction model using problem domain signatures and ticket routing sequences. In particular, it first applies supervised learning algorithms to select a set of potential resolver groups. Then, the experts are chosen based on their derived efficiency scores. Miao et al. (2010) presented a generative approach to recommend ticket routing decisions for new tickets in a network of expert groups, which considers the capability of expert groups either in resolving the tickets or in transferring the tickets to a resolver. Palshikar et al. (2011) proposed statistics-based discovery algorithms to identify experts for monitoring tickets, considering experts' efficiency in handling tickets of the specific problem. As an improvement of the collaborative network based methods (Miao et al., 2012; Sun et al., 2014; Xu and He, 2018) exploited an expert collaboration network model combining various information (expert profiles, problem descriptions, and resolution sequences). Furthermore, they developed a two-stage expert recommendation algorithm to determine a resolver for a ticket. In addition, Xu et al. (2018a) proposed three routing algorithms by mining ticket descriptions and resolution sequences from the historical resolved tickets. The first algorithm ranks all expert groups by computing a score between a ticket and a group profile to recommend an expert group with the highest score. The second and the third algorithms first assign an initial expert group for a ticket. Then, the second algorithm routes it to the next group with the highest transfer score among the neighbors. The third algorithm routes it to the next group by considering the combination of the group profile and the transfer profile. As the third one is globally optimized, it should perform better than the first two.

In contrast to the supervised methods above, Botezatu et al. (2015) used an unsupervised learning method to handle ticket triage tasks. More specifically, they first proposed a multi-view clustering method to cluster the tickets into categories that both reflect the ticketed problems and are homogeneous in the duration time for an expert to solve them during an expert's solving process. Then they devised a data-informed policy that assigns an incoming ticket to the expert who was the fastest in resolving it.

In addition, Zeng et al. (2017) improved their method in Zeng et al. (2014), which regarded ticket triage as a hierarchical classification problem. Specifically, they proposed a knowledge-guided hierarchical multi-label classification method to classify the monitoring tickets by taking the domain expertise into account and integrating it with hierarchical multi-label classification inference.

Some recent works (Han and Sun, 2020; Chen et al., 2019b) use deep learning for ticket triage. Han and Sun (2020) proposed DeepRouting, a multi-view deep neural network approach, to apply classic text matching and graph embedding for matching a ticket with the responsible group. DeepRouting utilizes the neural network models to automatically extract features from ticket content. Chen et al. (2019b) pointed out that the significant challenges of ticket reassignment are how to learn knowledge from incremental discussions and how to reduce the impact of noise from manual conversations. To solve these two challenges, they proposed DeepCT, a GRU-based model with a revised loss function and an attention-based mask strategy. DeepCT takes the following data as its input: the incident attributes (titles and summaries), environment information (monitor ID, related devices, and how the incidents are reported), and the discussions (textual information written by engineers in an incremental manner).

Furthermore, Wang et al. (2021) focused on the cross-service incident triage problem in large-scale cloud computing platforms. They proposed COT, the first incident triage approach that considers the global view of service correlations. COT mines the correlations among services from incident diagnosis data and infers the root cause of emerging incidents based on machine learning algorithms.

Summary. Content-based methods use ticket text to identify the responsible expert and focus on the one-time ticket assignment. Sequence-based methods use routing sequences of tickets to dispatch tickets, focusing on the ticket reassignment. Hybrid methods are generally more effective in handling both assignment and reassignment tasks by combining comprehensive ticket information, such as ticket texts, routing sequences, and experts' abilities.

5.3.2. Auxiliary methods for helping with manual triage

Although the IT service industry is shifting from a people-led and technology-assisted model into a people-assisted and technology-led model (Meng et al., 2018), it still requires manual participation when automated methods are unavailable or work bad. Automated triage methods can automatically identify responsible experts to complete triage. In addition, some methods do not automatically complete triage but provide auxiliary information to help OCEs make triage decisions. As mentioned earlier, incident linking (Section 5.2) can provide OCEs with more intuitive information to help them dispatch tickets. Moreover, some works named *ticket-based problem identification* can automatically identify the problems of tickets based on ticket analysis. Then OCEs can find responsible experts through identified problems. The identified problems can also help experts to mitigate and resolve incidents.

Diao et al. (2009) proposed a rule-based approach for ticket problem classification, collecting classification rules by crowd-sourcing. Experts can author rules, as well as socialize and execute rules through social networking. Potharaju et al. (2013) presented a method to infer problems, troubleshooting activities, and resolution actions for network tickets by automatically analyzing ticket texts based on the combination of statistical NLP, knowledge representation, and ontology modeling.

Bogojeska et al. (2013) proposed an automated approach to identify and rank the problematic servers based on server configurations and incident tickets. Specifically, they utilized a random forest classifier to predict whether the number of incident tickets for a given server exceeds a predefined threshold (i.e., the server is problematic) based on corresponding hardware, OS, and utilization information. In addition, they Bogojeska et al. (2014) utilized the gradient boosting machine for automatic server incident classification based on a small manually labeled subset of numerous incident tickets.

Shimpi et al. (2014) proposed two different algorithms to extract problems (snippets in the ticket description) from monitoring tickets and manual tickets, considering their inherent difference in structure and heterogeneity of text. Specifically, they proposed a clustering-based technique for monitoring tickets and a keyword discovery based approach for manual tickets. Then the identified problems can provide more concise and effective information to help make triage decisions. Xu et al. (2018b) proposed a ticket classification framework to identify problem types of monitoring/manual tickets automatically. First, they developed a ticket partition and signature construction method. The method integrates the domain knowledge extracted from historical tickets (e.g., the to-keep words, the to-discard words, and the synonym library) to improve accuracy and applies a local search strategy to construct ticket groups and signatures simultaneously. Then, a signature-based ticket classification algorithm was proposed to identify the problem type of an incoming ticket by finding the group with the most similar signature. A ticket will be delivered to maintenance teams for a manual classification when its similarity with the signature of any group is below a similarity threshold.

5.4. Incident mitigation

After a ticket is assigned, experts shall mitigate the incident quickly. Traditionally, experts try different mitigation actions (e.g., reboot the server, replace the faulty hardware, or expand the hardware capacity) based on their domain knowledge and analysis. However, manual analysis and mitigation exploration consume much time and prolong incident recovery time. Thus, some researchers are committed to speeding up the mitigation process.

In some organizations, experts record practical mitigation steps for specific incidents. For example, experts of Microsoft record documents describing mitigation processes named troubleshooting guides (TSGs) (Jiang et al., 2020). Jiang et al. (2020) found incidents could occur repeatedly and TSGs could be reused to facilitate incident mitigation. Therefore, they proposed an automated TSG recommendation approach, DeepRmd, leveraging the textual similarity between incident description and its corresponding TSG based on deep learning techniques. Intuitively, some incident linking methods in Section 5.2 can also find the best TSGs for current incidents by linking them with similar historical incidents.

Furthermore, some works intend to provide automated solutions for incident mitigation. Wu et al. (2012) proposed NetPilot, a system to automatically mitigate data center network failures by performing simple actions, such as deactivating or restarting suspected failure components. NetPilot circumvents the need for finding the exact root cause of a failure by taking an intelligent trial-and-error approach. NetPilot identifies a set of components that are likely to cause a problem and iteratively takes mitigation actions targeting each one until the problem is alleviated. The core of NetPilot comprises an impact estimator that helps prevent overly disruptive mitigation actions and a failure-specific mitigation planner that minimizes the number of trials. Wang et al. (2018b) defined and formalized the automation recommendation procedure as a multiarmed bandit problem with dependent arms. They proposed an intelligent system called AISTAR to achieve automated mitigation. AISTAR can promptly suggest and execute the most matched automation, a script prepared for the specific type of incidents using domain knowledge, mainly based on ticket content analysis. Deb et al. (2017) presented AESOP, a data-driven intelligent system to automatically learn policies and rules for triggering remedial actions for mitigating network service incidents. AESOP combines best operational practices with multiple measurement data, including vast numbers of logs capturing the operation actions and high-dimensional measurement time-series data capturing the conditions as these actions are performed.

5.5. Incident resolution

After an incident is successfully mitigated, experts need to take measures to thoroughly resolve the incident. On the one hand, experts need to find out the root cause of the incident. Related works have been introduced as background in Section 2.3. On the other hand, similar incidents could occur repetitively. As a result, the resolutions described in similar historical tickets can help find out the root cause and resolve the current incident (Tang et al., 2013a; Zhou et al., 2015b, 2016).

5.5.1. Resolution recommendation

Some incident linking techniques described in Section 5.2 and mitigation recommendation techniques in Section 5.4 can help recommend resolutions. In addition, Literature (Li et al., 2017) proposed the prospect of using recommender system techniques (Adomavicius and Tuzhilin, 2005) to address this problem. However, most recommender system techniques, especially those based on user responses, have not been applied to the resolution recommendation. Next, we will introduce some practical works on the resolution recommendation.

Tang et al. (2013a) proposed resolution recommendation algorithms by extending the k-nearest neighbor (KNN) algorithm (Altman, 1992). Resolutions of historical tickets with top summary similarity scores are recommended for new tickets. The similarity scores are calculated based on Jaccard similarity function after tokenizing each summary into a bag of words. However, Jaccard similarity has two main problems. First, it may ignore semantic information on tickets, which is important for similarity analysis. Second, the ticket summaries and resolutions may be noisy, especially for manual tickets, so the non-informative words lead to a low Jaccard similarity score of tickets with similar resolutions, making Jaccard similarity function inappropriate (Wang et al., 2017).

Some works use several techniques to alleviate the first problem by utilizing semantic information for the similarity measure. Kang et al. (2010) presented a knowledge-rich similarity measure to automatically discover the most similar historical incidents for a new incident. This measure incorporates three types of information, i.e., incident description, expert groups, and incident classification, to improve the capability of similarity measurement. Moreover, this measure exploits as much semantic knowledge as possible about various information contained in previous incidents. Zhou et al. (2015b, 2016) proposed two approaches to improve the method in Tang et al. (2013a). On the one hand, considering the resolutions often contain important information about incidents, they proposed a feature extraction approach capable of representing both the descriptions and resolutions using topic-level features obtained via the LDA model. On the other hand, considering that the effectiveness of KNN heavily relies on the underlying similarity measurement, they improved similarity measurement using metric learning. Zhou et al. (2015a) found out that vocabularies used in ticket descriptions are changing and shifting over time, but interesting mappings exist in those vocabularies. Then, they proposed a ticket resolution recommendation approach that accommodates vocabularies changing or shifting with time, based on a structural correspondence learning (SCL) domain adaptation algorithm.

The aforementioned methods deal with only semantically similar words without handling the noise caused by non-informative words. To address the second problem, Wang et al. (2017) presented a domain-specific approach to extract useful phrases to improve the resolution recommendation. The approach is based on ontology modeling that can enhance the semantic understanding of tickets and de-noise tickets by filtering non-informative words.

Furthermore, Zhou et al. (2017) developed an integrated framework to recommend resolutions efficiently. Considering the resolution may be a meaningless description, such as “no action”, the framework first uses a regression model to quantify the quality of resolutions. Then, it uses the tickets and their quality scores to train a deep neural network ranking model, which outputs the matching scores of ticket summary and resolution pairs. Finally, a resolution with the highest matching score is recommended.

5.5.2. Action recommendation

As the resolutions are usually verbose and cannot provide specific actions required for automation, some works recommend brief actions extracted from resolutions for rapid resolution of incidents.

Wei et al. (2007) proposed a technique to structure tickets for extracting resolution steps, applying the CRFs supervised learning process to identify individual units of information in the raw data. However, it relies on manual annotation of historical data to train a model based on CRFs. To achieve a more automated system, Aggarwal et al. (2016) presented ReAct, which can automatically identify a set of actions and the possible action sequence to resolve the issue mentioned in the ticket. The framework uses unstructured text analysis on historical tickets to find the next best actions and uses visualization to help users choose the most suitable options.

Agarwal et al. (2017) proposed a more intelligent system for automatic problem analysis and resolution for tickets. The system mainly includes three processes, i.e., identifying predefined problem classes for

tickets, mining problem linkages to recent system changes, and recommending resolution actions for resolving problems. More specifically, firstly, it uses the SVM method with a RBF Kernel to build a classifier to identify the problem classes of tickets. Secondly, it extracts features that link problems and changes in the historical data, and builds a machine learning model using these features, which, given a new problem, can predict whether and what type of change could have caused it. Finally, it uses NLP techniques, similarity analysis, and some rules to extract and recommend resolution actions that have been performed for issues corresponding to tickets pertaining to the same cause.

5.6. Other works

5.6.1. Empirical studies

In addition to the empirical studies mentioned early, there are some other empirical studies on incident ticket analysis. Bogojeska et al. (2014) focused on assessing the impact of hardware and OS currency on server availability. They conducted the data analysis to inspect the impact and importance of different server attributes (i.e., hardware and OS type, hardware age, OS currency, as well as CPU and memory utilization) on the rates of server failures based on a large set of incident tickets and server attribute data. Then they summarized the key conclusions, e.g., the hardware type and age highly impact server unavailability. These findings can further be used to derive guidelines for technology refresh decisions of the data center.

Giurgiu et al. (2014) conducted a large-scale study to analyze the factors affecting the labor effort necessary to solve server incidents. The factors include various incident features (i.e., the root cause, severity and complexity of the problem, the assigned support team, and the time shift when the incident is being resolved) and corresponding server features (i.e., the machine age and architecture, as well as the OS type and currency). The results show that the nature of the incidents and their complexity, the assigned support groups, as well as the underlying OS type play a major role in how much labor effort is spent on resolving such tickets.

Lou et al. (2013, 2017) shared their experience on IcM of a large-scale online service in Microsoft, mainly including using software analytics to solve engineers' pain points, the developed data-analysis techniques, and the lessons learned from research development and technology transfer. They also developed an industrial system based on a set of data-driven techniques, addressing several significant challenges in IcM practice, such as large-volume data, complex problem space, and incomplete domain knowledge.

Some works of Microsoft have analyzed incident characteristics from various perspectives. Chen et al. (2020a) analyzed incidents from 18 online service systems and found that many incidents are incidental incidents, which do not matter and do not need to be fixed with a high priority. Moreover, the incidental incidents are large-volume and cost much effort. So it is crucial to identify incidental incidents for IcM effort reduction. Accordingly, they proposed a deep learning based approach to prioritize incidents and find incidental incidents. Chen et al. (2020c) analyzed incident tickets from six core services in Microsoft, divided into four priorities (i.e., low, medium, high, and critical). They found that (1) incidents with lower priority (low and medium) are much more than incidents with higher priority (high and critical); (2) high-layer services (i.e., compute, database, and web service) may have hierarchical root causes, which increases problem search space and incident fixing time; (3) network issue with hardware failure and human error with code defect are the two major root causes for all incidents.

From the above descriptions, we can see that Microsoft usually treats all alerts as incidents, so most of their works are claimed to focus on analyzing incidents but not alerts, as shown in Table 4. In contrast, this paper separates alert management and incident management. On the one hand, it provides a more intuitive perspective to present each process and related works. On the other hand, it can reduce incident analysis effort more effectively by handling alerts using methods in alert management.

5.6.2. Ticket mining

There are some other works on ticket mining. Xu et al. (2016) identified the system situation related tickets from manual tickets based on SVMs ensemble to find missed monitoring tickets for further optimizing IcM systems. To understand where and how much effort is spent to resolve incidents, Branch et al. (2014) proposed a method to predict the service delivery effort by modeling the correlation between server characteristics (e.g., CPU type, OS type, memory size) and ticket properties (e.g., severity, creation time, close time) based on a support vector regression model.

Shetty et al. (2021) focused on structured knowledge extraction from service incidents. They proposed SoftNER, a deep learning based unsupervised framework for knowledge extraction from incidents, incorporating a novel multi-task BiLSTM-CRF model for named entity recognition. Furthermore, Shetty et al. (2022) extended the SoftNER framework by proposing an unsupervised approach to further extract entity relations and by constructing knowledge graphs using mutual information and co-occurrences. The knowledge extracted by SoftNER can be used to build accurate models for applications such as incident triage and entity recommendation based on their relevance to incident titles.

6. Limitation

First, this survey only provides a qualitative overview of AIM. A quantitative overview may provide more practical guidelines for practitioners. However, publicly available benchmark datasets and uniform evaluation approaches are needed to achieve quantitative comparison, which is missing in the literature. This can be considered a major future work.

Second, this survey does not follow a strict Systematic Literature Review (SLR) (Xiao and Watson, 2019), which may include research string, inclusion criteria, exclusion criteria, quality assessment, etc. Instead, we collect literature in a snowballing manner (Wohlin, 2014), which is known to be sensitive to the initial seeds. To alleviate this bias, we perform a keywords-based database search (MacDonell et al., 2010) as a complementary. However, the Keywords may be challenging to cover various related works. Besides, the literature screening is mainly based on the analysis and judgment of the first author. Therefore, the literature search may not be very comprehensive. We are concerned that some important literature is missing and will be sorry about this. In addition, as AIM is already a practice in the industry, unpublished but publicly available technical reports can also be valuable, named grey literature in SLR. In the future, we plan to conduct a Multivocal Literature Review (MLR) (Rijal et al., 2022), a form of SLR that includes the grey literature (e.g., blog posts and white papers), to improve the representativeness of selected papers.

Third, we may overlook some related but not so-related fields, such as event management (Li et al., 2017) (e.g., event correlation (Kotenko et al., 2022)), network security alert management (Mirheidari et al., 2013), and ITSM (Kubiak and Rass, 2018). As these fields may be beyond alert and incident management, we only focus on those works based on the analysis of alerts or incidents.

7. Future work

To provide practitioners and researchers with a deeper understanding of current situations and potential future directions, we summarize several main challenges and propose trend analysis for promising directions.

Table 7
How existing works address the challenges.

Challenge	Reference	Resolution
1	Section 4.1	Correlate related alerts to reduce analysis effort.
	Section 4.2	Use methods such as alert sequence analysis or alert summary to analyze current problems.
2	Section 4.3	Analyze alert authenticity or severity, or identify major incidents in alerts.
3	Section 5	Extract structured information from unstructured free text for processing equally, or propose customized methods for separately processing structured and unstructured text.
4	Section 5.2	Link incidents with related incidents or other information to speed up analysis.
	Section 5.3	Find responsible experts for incidents automatically, or provide auxiliary information to help find responsible experts manually.
	Section 5.4	Use mitigation information from historically similar incidents or automated methods to mitigate incidents.
	Section 5.5	Use resolution information or actions from historically similar incidents to resolve incidents.

7.1. Main challenges

We summarize several major challenges in AIM as follows.

1. **Large-volume and multi-source alerts.** As shown in Fig. 1, large-scale IT service systems are complicated because of numerous components and complex logical relationships (Zhao et al., 2020c). Alerts are multi-source as they can come from various indicators of each component at different levels. Moreover, a problem of an individual component may affect a series of components and their indicators (Jiang et al., 2011). Therefore, the number of alerts may be enormous and far exceed what engineers can investigate manually (Tang et al., 2012). Thus, it is challenging to analyze alerts.
2. **Many false alerts and missed alerts.** As service changes and business fluctuations are frequent in modern service systems, alerting rules formulated manually by engineers cannot fit continuously changing situations, causing the reporting of many false alerts and the missing of critical alerts (Tang et al., 2013c). False alerts cost much manual effort to be properly analyzed and handled, and missed alerts may cause ignoring some important incidents (Xu et al., 2016). Thus, false alerts and missed alerts make it challenging to find and resolve incidents in time.
3. **Multi-source incident tickets.** As mentioned earlier, there are two primary sources of incident tickets: monitoring tickets with semi-structured text from monitoring systems and manual tickets with unstructured free text from user complaints (Shimpi et al., 2014). This heterogeneity of incident tickets makes ticket analysis more challenging.
4. **Complex incident handling environments.** Due to the complexity of services, multiple expert groups are needed to maintain different service components (Chen et al., 2019b), such as networks, databases, and middleware. Finding responsible experts for incidents is cumbersome and is not conducive to rapidly resolving incidents (Han and Sun, 2020). In addition, the variety of incidents and complex root causes make it challenging to resolve incidents.

To address the challenges above, various intelligent AIM technologies have been widely studied. Almost all the existing research aims to address at least one of the four challenges, as shown in Table 7. Alert correlation (Section 4.1) and alert storm handling (Section 4.2) response to challenge 1. Alert determination (Section 4.3) aims to handle challenge 2. Many works in Section 5 consider the heterogeneity of monitoring tickets and manual tickets (challenge 3) and provide corresponding solutions. Incident Triage (Section 5.3) helps find responsible experts for tickets to cope with a problem in challenge 4. Incident linking (Section 5.2), incident mitigation (Section 5.4), and incident resolution (Section 5.5) separately aim to speed up incident analysis, mitigation, or resolution (response to challenge 4). In addition, the experience of some other works (Section 5.6) can also help improve service systems or resolve incidents (response to challenge 4).

7.2. Automation analysis

Although existing works have made a certain degree of intelligent improvement in AIM, many intermediate processes still require manual intervention, which needs to be pointed out and understood. First, almost any alert analysis work requires data cleaning or preprocessing to make the data format suitable for customized solutions, which is a fundamental step that requires human intervention. Besides, the works based on alert attribute analysis generally require feature selection from many alert features, which requires manual analysis or conducting feature engineering by relevant personnel.

Furthermore, although we concatenate the existing individual works of each module in our architecture (Fig. 2), these works still require human intervention. Alert correlation is a significant improvement in reducing manual analysis pressure. However, the results of alert correlation still need manual analysis to determine whether/how to create tickets. Although alert storm processing or alert determination can further identify critical alerts or possible incidents, ticket creation still depends on cautious manual analysis due to the labor-intensive processing for each ticket.

Incident management after ticket creation still requires human intervention. First, a customized ticket representation strategy is required before any intelligent solution can be used. For manual tickets in particular, while some solutions try to maximize their effectiveness, analyzing manual tickets may still require human involvement. Ticket linking is an alert correlation-like step that aims to help OCEs or experts with ticket triage and analysis tasks. Therefore, linked tickets still need to be viewed manually. Two types of methods are introduced in ticket triage: automated triage methods, which can automatically identify the responsible expert (group) for each ticket, and auxiliary methods, which extract knowledge from the ticket to guide the triage. Clearly, auxiliary methods require human decisions. But in practice, automated methods may still require human intervention as they struggle to cope with various tickets in complex IT service environments.

For incident mitigation or resolution, current intelligent solutions can only recommend historical solutions of similar incidents for reference or perform simple operations (such as restarts) in an attempt to fix the incident. Therefore, these processes still require manual intervention, especially for complex incidents. After successfully resolving an incident, reviewing the accident, including the root cause and solution summary, requires a lot of manual effort.

7.3. Potential directions

Although existing works have alleviated the challenges to some extent, some directions can still be explored or optimized. First, as mentioned in Section 6, we need publicly available benchmark datasets and unified evaluation approaches for a quantitative overview in the future. Moreover, some other urgent directions for intelligent AIM technologies are as follows.

7.3.1. Optimize alert generation

Improve alert generation based on the analysis results of alerts and tickets. False/missed alerts, which cost much manual effort to handle, are widespread in daily ITSM. On the one hand, it is because alert rules may not dynamically fit frequent fluctuations in the usage of IT services. On the other hand, originally suitable methods may be inapplicable as the service systems evolve. Thus, the alert generation needs to be continuously optimized.

As false/missed alerts often indicate inappropriate alerting rules, optimizing alerting generation based on the analysis results of alerts and tickets is very practical and valuable. But existing works mainly focus on analyzing alerts after the alerts are generated. Only a few works focus on optimizing alerting rules based on alert and incident analysis. Tang et al. (2012) optimized alerting rules based on the offline analysis of historical alerts and the matching tickets, where potential monitoring conditions are built on a set of predictive rules automatically generated by a rule-based learning algorithm with coverage, confidence, and rule complexity criteria. They also presented an integrated framework to reduce false and missed alerts for an automatic monitoring system (Tang et al., 2013c), mainly based on the finding that many false-positive alerts were transient and could be cleared automatically by waiting for some minutes.

In fact, some existing techniques already provide support to optimize alerting rules. For example, some alert determination methods in Section 4.3 can identify false/non-severe alerts, which can be further analyzed to optimize alerting rules. Identifying incidental incidents (Chen et al., 2020a) can help reduce false-positive alerts. Discovering manual tickets not reported by the monitoring system (Xu et al., 2016) can help reduce missed alerts. Tang et al. (2013b) apply a text classification model to analyze manual ticket descriptions and identify the corresponding system issues, aiming to identify missed monitoring alerts based on manual tickets. Moreover, some frequent or periodic false alerts can be found by combining time sequence analysis methods (Elfeky et al., 2005; Puech et al., 2019). We advocate for more intelligent optimization algorithms for learning and dynamically optimizing alerting rules based on multiple-dimension alert and incident analysis results.

Apply and optimize the whitelist mechanism. In many business-sensitive services, their monitoring systems report more alerts to avoid missing possible failures. Some alerts are reported continuously and identified as false alerts. Nevertheless, these alerts cannot be permanently deleted as there is no guarantee that these alerts will not be failures at another time. Except for optimizing alerting rules, some organizations enable the whitelist mechanism to solve this problem by adding these alerts to the whitelist. Alerts in the whitelist will be stopped reporting for some time. Thus, reasonable use and optimization of the whitelist mechanism can also effectively reduce the effort in alert management. This is a valuable direction with little research and deserves more attention.

Provide richer standardized fields for alerts or incidents. Although some monitoring systems provide rich and detailed descriptive fields for each alert or incident, many do not offer these useful fields, especially for those manually generated incidents. Providing more useful fields is valuable for subsequent analysis, especially when applying some automated analysis methods. Some NLP techniques (e.g., keyphrase extraction (Papagiannopoulou and Tsoumakas, 2020)) provide solutions to automatically extract valuable fields (such as alert components and indicators) from complex alert descriptions.

7.3.2. Deal with alert storms

As mentioned in Section 4.2, alert storms have become a serious problem in daily alert management. But currently, only a few research papers (Zhao et al., 2020a; Li et al., 2022c) focus on handling IT service alert storms. As a significant challenge in IT service alert management, the alert storm problem deserves more attention and research.

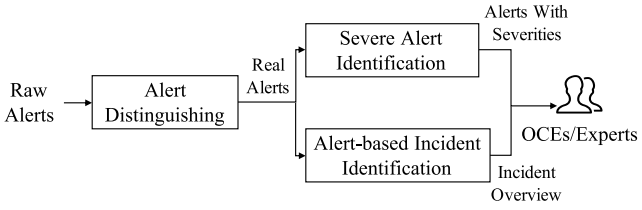


Fig. 7. Integrated process of three alert determination methods.

Utilize deep learning techniques to integrate multiple kinds of information. As shown in Fig. 1, there are many service components in modern IT service systems, and their dependencies are very complex. Various information can be used to analyze and handle alert storms, such as various dependencies (e.g., deployment dependencies, topological relationships, and causalities), alert contents, and alert sequences. Multiple kinds of information is usually difficult to integrate to analyze and deal with problems. However, deep learning techniques have shown effectiveness in integrating multiple kinds of information to solve problems. Thus, utilizing deep learning techniques to integrate multiple kinds of information can be a promising direction to handle alert storms and also some other problems. More specifically, a deep model can be trained to identify the root causes of alert storms and speed up their handling. Nevertheless, it is often necessary to accumulate a large number of cases as a training dataset.

Combine alert correlation and alert storm handling techniques. Currently, engineers usually adopt a triggered way to deal with critical alert storm problems in real time. However, streaming analysis has some advantages over the passive triggering defense mechanism, e.g., apportioning computation also into the period without SLA violations and little time cost after SLA violations. Some works on alert correlation (Section 4.1) conduct streaming analysis to solve the large volume of raw alerts and insufficient information from a single alert. Applying similar streaming analysis techniques in alert storm handling is promising. Although alert correlation can effectively alleviate alert storms, alert storms may still occur when facing extreme failure situations or the effect of applied alert correlation technologies is not good. Providing a triggered way to deal with these extreme situations is necessary and complementary. Thus, combining the alert correlation techniques and the alert storm handling techniques can provide both the active streaming alert correlation and the passive triggering defense mechanism against alert storms. This combination can achieve more effective alert processing results to significantly reduce the effort of manually analyzing and processing alerts.

Improve the effectiveness of alert determination by combining three types of methods. As mentioned in Section 4.3.4, the three methods of alert determination (i.e., alert distinguishing, severe alert identification, and alert-based incident identification) have their own advantages and disadvantages. Better results can be achieved by giving full play to the advantages of each method and making up for their shortcomings. Fig. 7 shows an integrated process of three alert determination methods. First, alert distinguishing can be applied to select real alerts and move out false alerts from alerts to be analyzed. Then, these real alerts can be analyzed by both severe alert identification and alert-based incident identification. The former can sort the alerts by their severity, and the latter can provide an overview of current incidents. The combination can provide OCEs/experts with an overview of current incidents and severe alerts with their severities, which can exploit the advantages of the latter while avoiding missing important alert information. At the same time, alert distinguishing has filtered out many false alerts to reduce analysis pressure at the first stage.

7.3.3. Improve incident analysis

Improve incident triage by providing automated triage and auxiliary information. In automated triage methods, hybrid methods have been shown to be more effective and promising as they combine comprehensive information and optimize the triage results from multiple angles. With the ticket information recorded in the incident database becoming more comprehensive, using deep learning methods to combine various information to achieve automated incident triage is a promising direction. At present, although automated triage methods have been widely studied, manual analysis is still essential in practice since automated triage methods cannot always be effective. So auxiliary methods for helping with manual triage are still crucial. Therefore, it is a promising direction to integrate automated and auxiliary methods. This can achieve a complete incident triage system by both automating incident triage and providing auxiliary information to help with manual triage.

Automatize incident mitigation and incident resolution. In current IcM, experts often manually mitigate and resolve incidents based on their domain knowledge. Although resolution recommendation can help resolve current incidents, completing incident mitigation and resolution still costs much manual effort. Action recommendation steps forward with more intelligent suggestions. While executing scripts created by experts achieves automatic mitigation in certain scenarios (Wang et al., 2018b). More methods with more intelligence are still in need to reduce human effort and accelerate incident mitigation and resolution.

7.3.4. Develop/improve AIOps framework for AIM

AIOps, as one of the most important developments in service assurance in the last few years, has been commonly used in the transformation of modern service management systems. In general, most modern AIM follow the AIOps strategies. As a result, some AIOps frameworks (Chen et al., 2020c; Li et al., 2022c; Gaikwad et al., 2021) for AIM have been proposed in the literature. Besides, various AIOps products (e.g., moogsoft[13]) have been used in daily IT service operations. They use many AIOps techniques to improve AIM performance. In contrast, the techniques in our architecture seem to be the most systematic and comprehensive. Our architecture can help improve the processing flow of the AIOps framework, and the technologies in each process can help improve the corresponding module. We believe this survey can facilitate the development of AIOps AIM systems.

Besides, some works (e.g., Chen et al., 2022a; Arain et al., 2022) focus on the efficiency of collaboration between operations roles because multiple roles are required to view and process alerts and incidents for large-scale IT service systems. These roles may have different duty times, current workloads, abilities, experiences, etc. Therefore, the optimization of role collaboration is required for the operations of large-scale service systems. But the related works are not much introduced in our survey.

7.4. Related technologies

This paper is a survey of research papers on the topic of AIM, focusing on related algorithms. Relevant papers rarely introduce details or specific techniques in engineering practice. Therefore, we present some insights into related technologies based on practical experience in our real-world AIOps.

7.4.1. Programming languages

In AIOps practice, the most commonly used language for developing AIM tools and systems should be Java which has a complete ecosystem. Some important applications, such as Flink,[14] are mainly developed based on the Java language. Besides, the Scala language is another widely used language for development practice these years. Scala has been used in important applications, such as Spark,[15] a big data processing framework, and Kafka,[16] a distributed publish/subscribe system. Scala is suitable for the development of big data, data mining, NLP, machine learning, etc. These two languages are widely used because large-scale streaming processing is essential in alert processing. Mixed programming development in Java and Scala would be an excellent choice for developers.

In addition, in related research fields, the Python language is widely used to implement various algorithm prototypes quickly as it is easy to understand and learn and has many algorithm libraries.

7.4.2. Open-source tools

In AIOps practice, several open-source tools can be used to achieve various goals. Grafana[17] is an open-source data visualization tool with rich dashboards and charts. It can visualize various data and support multiple data sources, such as Graphite, InfluxDB, Elasticsearch, Mysql, etc. Grafana allows users to configure thresholds and conditions for receiving alerts. And alert notifications can be received through PagerDuty, Slack, Webhooks, and Gmail. Therefore, Grafana can be used to display various metrics, alerts, incidents, etc.

Some system monitoring tools, such as Prometheus[18] and Zabbix, can be exploited. The main difference between them is that the database used by Prometheus is the time series database TSDB, and the database used by Zabbix is the relational database of MySQL or PgSQL. Therefore, Prometheus is more suitable for time series data such as metrics, and Zabbix is more suitable for storing data such as logs.

Flink is a framework and distributed processing engine for stateful computations over unbounded and bounded data streams. It is very accurate in data ingestion and can easily recover from failures while maintaining the state. Flink can be used with Kafka to transmit metrics, alerts, and other data efficiently.

In our current AIM practices, we use big data components (e.g., Flink) to generate, process, and distribute alerts in real time. However, we rarely use big data or high-performance computing components when conducting AIOps alert analyses (e.g., alert storm analysis). This is because the amount of alerts and incidents is relatively small compared to various monitoring metrics. However, as the scale of the service system continues to expand, the volume of alerts and incidents may exceed the current single-machine analysis capability. Therefore, high-performance computing and big data components (e.g., Hadoop[19] and Spark) can be widely used in future alert and incident analysis.

8. Conclusion

This paper has reviewed the research works carried out in intelligent AIM in IT services. First of all, for a unified description and better understanding, the paper summarizes an architecture of AIM. Then, a detailed literature review is conducted based on corresponding processes in the architecture. Finally, the paper analyzes the current challenges and trends in AIM.

To the best of our knowledge, this is the most comprehensive survey on systematic AIM in IT services. This survey can help newcomers understand this field, help researchers conduct relevant research, and help

[13] https://www.moogsoft.com.

[14] https://flink.apache.org.
[15] https://spark.apache.org.
[16] https://kafka.apache.org.
[17] https://grafana.com.
[18] https://prometheus.io.
[19] https://hadoop.apache.org.

industrial organizations design and improve the management systems from each process. We believe our survey can help drive the evolution and development of AIOps AIM systems, help ensure the quality of real-world IT services, and help improve citizens' experience of the IT services.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported by the National Key R&D Program of China 2019YFB1802504, and the State Key Program of National Natural Science of China under Grant 62072264.

References

Ab Rahman, N.H., Choo, K.-K.R., 2015. A survey of information security incident handling in the cloud. Comput. Secur. 49, 45–69.
Aceto, G., Botta, A., De Donato, W., Pescapè, A., 2013. Cloud monitoring: A survey. Comput. Netw. 57 (9), 2093–2115.
Adomavicius, G., Tuzhilin, A., 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17 (6), 734–749.
Agarwal, S., Aggarwal, V., Akula, A.R., Dasgupta, G.B., Sridhara, G., 2017. Automatic problem extraction and analysis from unstructured text in IT tickets. IBM J. Res. Dev. 61 (1), 4–41.
Agarwal, S., Sindhgatta, R., Sengupta, B., 2012. SmartDispatch: enabling efficient ticket dispatch in an IT service environment. In: SIGKDD. pp. 1393–1401.
Aggarwal, V., Agarwal, S., Dasgupta, G.B., Sridhara, G., Vijay, E., 2016. ReAct: a system for recommending actions for rapid resolution of IT service incidents. In: SCC. pp. 1–8.
Ahmed, K., Izadi, I., Chen, T., Joe, D., Burton, T., 2013. Similarity analysis of industrial alarm flood data. IEEE Trans. Autom. Sci. Eng. 10 (2), 452–457.
Ahmed, M., Mahmood, A.N., Hu, J., 2016. A survey of network anomaly detection techniques. J. Netw. Comput. Appl. 60, 19–31.
Aho, A.V., Corasick, M.J., 1975. Efficient string matching: an aid to bibliographic search. Commun. ACM 18 (6), 333–340.
Alhaj, T.A., Siraj, M.M., Zainal, A., Elshoush, H.T., Elhaj, F., 2016. Feature selection using information gain for improved structural-based alert correlation. PLoS One 11 (11), e0166017.
Altman, N.S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. Amer. Statist. 46 (3), 175–185.
Arain, T.A., Huang, X., Cai, Z., Xu, J., 2022. Multi-objective optimization of ticket assignment problem in large data centers. In: CCF Conference on Computer Supported Cooperative Work and Social Computing. Springer, pp. 37–51.
Arraj, V., 2010. ITIL®: The Basics. Buckinghampshire, UK.
Asres, M.W., Mengistu, M.A., Castrogiovanni, P., Bottaccioli, L., Macii, E., Patti, E., Acquaviva, A., 2020. Supporting telecommunication alarm management system with trouble ticket prediction. IEEE Trans. Ind. Inform. 17 (2), 1459–1469.
Blei, D.M., Ng, A.Y., Jordan, M.I., 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3, 993–1022.
Bogojeska, J., Giurgiu, I., Lanyi, D., Stark, G., Wiesmann, D., 2014. Impact of HW and OS type and currency on server availability derived from problem ticket analysis. In: NOMS. pp. 1–9.
Bogojeska, J., Lanyi, D., Giurgiu, I., Stark, G., Wiesmann, D., 2013. Classifying server behavior and predicting impact of modernization actions. In: CNSM. pp. 59–66.
Botezatu, M.M., Bogojeska, J., Giurgiu, I., Voelzer, H., Wiesmann, D., 2015. Multi-view incident ticket clustering for optimal ticket dispatching. In: SIGKDD. pp. 1711–1720.
Branch, J.W., Diao, Y., Shwartz, L., 2014. A framework for predicting service delivery efforts using IT infrastructure-to-incident correlation. In: NOMS. pp. 1–8.
Brewster, E., Griffiths, R., Lawes, A., Sansbury, J., 2012. IT Service Management: A Guide for ITIL Foundation Exam Candidates. BCS, The Chartered Institute for IT.
Carbonneau, M.-A., Cheplygina, V., Granger, E., Gagnon, G., 2018. Multiple instance learning: A survey of problem characteristics and applications. Pattern Recognit. 77, 329–353.
Chalapathy, R., Chawla, S., 2019. Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.
Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM Comput. Surv. 41 (3), 1–58.
Chandola, V., Banerjee, A., Kumar, V., 2010. Anomaly detection for discrete sequences: A survey. IEEE Trans. Knowl. Data Eng. 24 (5), 823–839.
Charbonnier, S., Bouchair, N., Gayet, P., 2016. Fault template extraction to assist operators during industrial alarm floods. Eng. Appl. Artif. Intell. 50, 32–44.
Chen, H., Duncklee, A., Jajodia, S., Liu, R., Mcnamara, S., Subrahmanian, V., 2022a. PCAM: A data-driven probabilistic cyber-alert management framework. ACM Trans. Internet Technol. (TOIT) 22 (3), 1–24.
Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. In: SIGKDD. pp. 785–794.
Chen, J., He, X., Lin, Q., Xu, Y., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y., Zhang, D., 2019a. An empirical investigation of incident triage for online service systems. In: ICSE-SEIP. pp. 111–120.
Chen, J., He, X., Lin, Q., Zhang, H., Hao, D., Gao, F., Xu, Z., Dang, Y., Zhang, D., 2019b. Continuous incident triage for large-scale online service systems. In: ASE. pp. 364–375.
Chen, Z., Kang, Y., Li, L., Zhang, X., Zhang, H., Xu, H., Zhou, Y., Yang, L., Sun, J., Xu, Z., et al., 2020c. Towards intelligent incident management: why we need it and how we make it. In: ESEC/FSE. pp. 1487–1497.
Chen, Z., Liu, J., Su, Y., Zhang, H., Wen, X., Ling, X., Yang, Y., Lyu, M.R., 2021. Graph-based incident aggregation for large-scale online service systems. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 430–442.
Chen, P., Qi, Y., Zheng, P., Hou, D., 2014. CauseInfer: Automatic and distributed performance diagnosis with hierarchical causality graph in large distributed systems. In: INFOCOM. pp. 1887–1895.
Chen, J., Wang, P., Wang, W., 2022b. Online summarizing alerts through semantic and behavior information. In: Proceedings of the 44th International Conference on Software Engineering. pp. 1646–1657.
Chen, Y., Yang, X., Dong, H., He, X., Zhang, H., Lin, Q., Chen, J., Zhao, P., Kang, Y., Gao, F., Xu, Z., Zhang, D., 2020b. Identifying linked incidents in large-scale online service systems. In: ESEC/FSE. pp. 304–314.
Chen, Y., Yang, X., Lin, Q., Zhang, H., Gao, F., Xu, Z., Dang, Y., Zhang, D., Dong, H., Xu, Y., et al., 2019c. Outage prediction and diagnosis for cloud service systems. In: WWW. pp. 2659–2665.
Chen, J., Zhang, S., He, X., Lin, Q., Zhang, H., Hao, D., Kang, Y., Gao, F., Xu, Z., Dang, Y., et al., 2020a. How incidental are the incidents? Characterizing and prioritizing incidents for large-scale online service systems. In: ASE. pp. 373–384.
Chengpo, M., Houkuan, H., Shengfeng, T., 2006. A survey of intrusion-detection alert aggregation and correlation techniques. J. Comput. Res. Dev. 43 (1), 1.
Costa, R., Cachulo, N., Cortez, P., 2009. An intelligent alarm management system for large-scale telecommunication companies. In: Portuguese Conference on Artificial Intelligence. Springer, pp. 386–399.
Dai, Z., Xiong, C., Callan, J., Liu, Z., 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In: WSDM. pp. 126–134.
Dang, Y., Lin, Q., Huang, P., 2019. AIOps: real-world challenges and research innovations. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings. ICSE-Companion, IEEE, pp. 4–5.
Dasgupta, G.B., Nayak, T.K., Akula, A.R., Agarwal, S., Nadgowda, S.J., 2014. Towards auto-remediation in services delivery: Context-based classification of noisy and unstructured tickets. In: ICSOC. pp. 478–485.
Deb, S., Ge, Z., Isukapalli, S., Puthenpura, S., Venkataraman, S., Yan, H., Yates, J., 2017. Aesop: Automatic policy learning for predicting and mitigating network service impairments. In: SIGKDD. pp. 1783–1792.
Derdack, M., 2017. Definition of event, alert, incident and notification. URL: https://www.linkedin.com/pulse/definition-event-alert-incident-notification-matthes-derdack/.
Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
Diao, Y., Jamjoom, H., Loewenstern, D., 2009. Rule-based problem classification in it service management. In: 2009 IEEE International Conference on Cloud Computing. pp. 221–228.
Diao, Y., Jan, E., Li, Y., Rosu, D., Sailer, A., 2016. Service analytics for IT service management. IBM J. Res. Dev. 60 (2–3), 13:1–13:17.
Du, M., Li, F., Zheng, G., Srikumar, V., 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: CCS. pp. 1285–1298.
Duan, Z., Zhang, Z.-L., Hou, Y.T., 2003. Service overlay networks: SLAs, QoS, and bandwidth provisioning. IEEE/ACM Trans. Netw. 11 (6), 870–883.
Elfeky, M.G., Aref, W.G., Elmagarmid, A.K., 2005. Periodicity detection in time series databases. IEEE Trans. Knowl. Data Eng. 17 (7), 875–887.
Elkahky, A.M., Song, Y., He, X., 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In: WWW. pp. 278–288.
Equipment, E., Association, M.U., 1999. Alarm Systems: A Guide To Design, Management and Procurement. Engineering Equipment and Materials Users Association London.
Eyerman, S., Eeckhout, L., 2008. System-level performance metrics for multiprogram workloads. IEEE Micro 28 (3), 42–53.

Fonseca, R., Porter, G., Katz, R.H., Shenker, S., 2007. X-trace: A pervasive network tracing framework. In: NSDI.

Fournier-Viger, P., He, G., Zhou, M., Nouioua, M., Liu, J., 2020. Discovering alarm correlation rules for network fault management. In: ICSOC. pp. 228–239.

Gaikwad, R., Deshpande, S., Vaidya, R., Bhate, M., 2021. A framework design for algorithmic it operations (aiops). Des. Eng. 2037, 2044.

Gan, Y., Liang, M., Dev, S., Lo, D., Delimitrou, C., 2021. Sage: practical and scalable ML-driven performance debugging in microservices. In: ASPLOS. pp. 135–151.

Gan, Y., Zhang, Y., Hu, K., Cheng, D., He, Y., Pancholi, M., Delimitrou, C., 2019. Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices. In: ASPLOS. pp. 19–33.

Gao, J., Pantel, P., Gamon, M., He, X., Deng, L., 2014. Modeling Interestingness with Deep Neural Networks. Technical Report MSR-TR-2014-56, URL: https://www.microsoft.com/en-us/research/publication/modeling-interestingness-with-deep-neural-networks/.

Giurgiu, I., Bogojeska, J., Nikolaiev, S., Stark, G., Wiesmann, D., 2014. Analysis of labor efforts and their impact factors to solve server incidents in datacenters. In: CCGRID. pp. 424–433.

Godbole, S., Roy, S., 2008. Text classification, business intelligence, and interactivity: automating c-sat analysis for services industry. In: SIGKDD. pp. 911–919.

González-Granadillo, G., González-Zarzosa, S., Diaz, R., 2021. Security information and event management (SIEM): analysis, trends, and usage in critical infrastructures. Sensors 21 (14), 4759.

Granger, C.W., 1969. Investigating causal relations by econometric models and cross-spectral methods. Econometrica 424–438.

Gu, J., Wen, J., Wang, Z., Zhao, P., Luo, C., Kang, Y., Zhou, Y., Yang, L., Sun, J., Xu, Z., et al., 2020. Efficient customer incident triage via linking with system incidents. In: ESEC/FSE. pp. 1296–1307.

Guo, C., Hu, W., Lai, S., Yang, F., Chen, T., 2017. An accelerated alignment method for analyzing time sequences of industrial alarm floods. J. Process Control 57, 102–115.

Gupta, R., Prasad, K.H., Luan, L., Rosu, D., Ward, C., 2009. Multi-dimensional knowledge integration for efficient incident management in a services cloud. In: SCC. pp. 57–64.

Gupta, R., Prasad, K.H., Mohania, M., 2008a. Automating ITSM incident management process. In: ICAC. pp. 141–150.

Gupta, R., Prasad, K.H., Mohania, M., 2008b. Information integration techniques to automate incident management. In: NOMS. pp. 979–982.

Han, J., Sun, A., 2020. DeepRouting: A deep neural network approach for ticket routing in expert network. In: SCC. pp. 386–393.

Harper, R., Tee, P., 2018. Cookbook, a recipe for fault localization. In: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium. IEEE, pp. 1–6.

Hassan, W.U., Guo, S., Li, D., Chen, Z., Jee, K., Li, Z., Bates, A., 2019. Nodoze: Combatting threat alert fatigue with automated provenance triage. In: Network and Distributed Systems Security Symposium.

He, S., Zhu, J., He, P., Lyu, M.R., 2016. Experience report: System log analysis for anomaly detection. In: ISSRE. pp. 207–218.

Hládek, D., Staš, J., Pleva, M., 2020. Survey of automatic spelling correction. Electronics 9 (10), 1670.

Hodge, V., Austin, J., 2004. A survey of outlier detection methodologies. Artif. Intell. Rev. 22 (2), 85–126.

Hu, W., Chen, T., Shah, S.L., 2018. Detection of frequent alarm patterns in industrial alarm floods using itemset mining methods. IEEE Trans. Ind. Electron. 65 (9), 7290–7300.

Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., Heck, L., 2013. Learning deep structured semantic models for web search using clickthrough data. In: CIKM. pp. 2333–2338.

International Society of Automation (ISA), 2009. Management of Alarm Systems for the Process Industries. No. ANISI-ISA-18.2-2009.

Jan, E.-E., Ni, J., Ge, N., Ayachitula, N., Zhang, X., 2013. A statistical machine learning approach for ticket mining in IT service delivery. In: IM. pp. 541–546.

Jiang, G., Chen, H., Yoshihira, K., Saxena, A., 2011. Ranking the importance of alerts for problem determination in large computer systems. Cluster Comput. 14 (3), 213–227.

Jiang, J., Lu, W., Chen, J., Lin, Q., Zhao, P., Kang, Y., Zhang, H., Xiong, Y., Gao, F., Xu, Z., et al., 2020. How to mitigate the incident? an effective troubleshooting guide recommendation technique for online service systems. In: ESEC/FSE. pp. 1410–1420.

Joachims, T., 1998. Text categorization with support vector machines: Learning with many relevant features. In: European Conference on Machine Learning. pp. 137–142.

Johnson, R., Zhang, T., 2017. Deep pyramid convolutional neural networks for text categorization. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 562–570.

Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T., 2016. Fasttext.zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

Kaldor, J., Mace, J., Bejda, M., Gao, E., Kuropatwa, W., O'Neill, J., Ong, K.W., Schaller, B., Shan, P., Viscomi, B., et al., 2017. Canopy: An end-to-end performance tracing and analysis system. In: SOSP. pp. 34–50.

Kalisch, M., Bühlman, P., 2007. Estimating high-dimensional directed acyclic graphs with the PC-algorithm. J. Mach. Learn. Res. 8 (3).

Kang, Y.-B., Zaslavsky, A., Krishnaswamy, S., Bartolini, C., 2010. A knowledge-rich similarity measure for improving IT incident resolution process. In: SAC. pp. 1781–1788.

Khan, A., Jamjoom, H., Sun, J., 2009. AIM-HI: a framework for request routing in large-scale IT global service delivery. IBM J. Res. Dev. 53 (6), 4–41.

Khan, S.S., Madden, M.G., 2009. A survey of recent trends in one class classification. In: Irish Conference on Artificial Intelligence and Cognitive Science. pp. 188–197.

Kim, S., Brown, E.N., 2010. A general statistical framework for assessing granger causality. In: IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 2222–2225.

Kim, M., Sumbaly, R., Shah, S., 2013. Root cause detection in a service-oriented architecture. ACM SIGMETRICS Perform. Eval. Rev. 41 (1), 93–104.

Kotenko, I., Gaifulina, D., Zelichenok, I., 2022. Systematic literature review of security event correlation methods. IEEE Access.

Krishnan, G., Ravindran, V., 2017. IT service management automation and its impact to IT industry. In: ICCIDS. pp. 1–4.

Kubiak, P., Rass, S., 2018. An overview of data-driven techniques for IT-service-management. IEEE Access 6, 63664–63688.

Kuhn, M., Johnson, K., 2019. Feature Engineering and Selection: A Practical Approach for Predictive Models. CRC Press.

Lai, S., Yang, F., Chen, T., 2017. Online pattern matching and prediction of incoming alarm floods. J. Process Control 56, 69–78.

Landauer, M., Skopik, F., Wurzenberger, M., Rauber, A., 2022. Dealing with security alert flooding: using machine learning for domain-independent alert aggregation. ACM Trans. Priv. Secur. 25 (3), 1–36.

Laptev, N., Amizadeh, S., Flint, I., 2015. Generic and scalable framework for automated time-series anomaly detection. In: SIGKDD. pp. 1939–1947.

Li, Z., Chen, J., Jiao, R., Zhao, N., Wang, Z., Zhang, S., Wu, Y., Jiang, L., Yan, L., Wang, Z., et al., 2021b. Practical root cause localization for microservice systems via trace analysis. In: IWQoS.

Li, M., Li, Z., Yin, K., Nie, X., Zhang, W., Sui, K., Pei, D., 2022a. Causal inference-based root cause analysis for online service systems with intervention recognition. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. arXiv:2206.05871.

Li, S., Qin, D., Wu, X., Li, J., Li, B., Han, W., 2022b. False alert detection based on deep learning and machine learning. Int. J. Semant. Web Inf. Syst. (IJSWIS) 18 (1), 1–21.

Li, T., Zeng, C., Jiang, Y., Zhou, W., Tang, L., Liu, Z., Huang, Y., 2017. Data-driven techniques in computing system management. ACM Comput. Surv. 50 (3), 1–43.

Li, Y., Zhang, X., He, S., Chen, Z., Kang, Y., Liu, J., Li, L., Dang, Y., Gao, F., Xu, Z., et al., 2022c. An intelligent framework for timely, accurate, and comprehensive cloud incident detection. Oper. Syst. Rev. 56 (1), 1–7.

Li, L., Zhang, X., Zhao, X., Zhao, P., Qiao, B., He, S., Lee, P., Sun, J., Gao, F., Yang, L., et al., 2021a. Fighting the fog of war: Automated incident detection for cloud systems. In: USENIX ATC. pp. 131–146.

Li, Z., Zhao, N., Li, M., Lu, X., Wang, L., Chang, D., Cao, L., Zhang, W., Sui, K., Wang, Y., Du, X., Duan, G., Pei, D., 2022d. Actionable and interpretable fault localization for recurring failures in online service systems. In: Proceedings of the 2022 30th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.

Lin, Y., Chen, Z., Cao, C., Tang, L.-A., Zhang, K., Cheng, W., Li, Z., 2018b. Collaborative alert ranking for anomaly detection. In: CIKM. pp. 1987–1995.

Lin, J., Chen, P., Zheng, Z., 2018a. Microscope: Pinpoint performance issues with causal graphs in micro-service environments. In: ICSOC. pp. 3–20.

Lin, S., Hong, W., Wang, D., Li, T., 2017. A survey on expert finding techniques. J. Intell. Inf. Syst. 49 (2), 255–279.

Lin, D., Raghu, R., Ramamurthy, V., Yu, J., Radhakrishnan, R., Fernandez, J., 2014. Unveiling clusters of events for alert and incident management in large-scale enterprise it. In: SIGKDD. pp. 1630–1639.

Liu, D., He, C., Peng, X., Lin, F., Zhang, C., Gong, S., Li, Z., Ou, J., Wu, Z., 2021. MicroHECL: High-efficient root cause localization in large-scale microservice systems. In: ICSE-SEIP. pp. 338–347.

Liu, P., Xu, H., Ouyang, Q., Jiao, R., Chen, Z., Zhang, S., Yang, J., Mo, L., Zeng, J., Xue, W., et al., 2020a. Unsupervised detection of microservice trace anomalies through service-level deep bayesian networks. In: ISSRE. pp. 48–58.

Liu, P., Zhang, S., Sun, Y., Meng, Y., Yang, J., Pei, D., 2020b. FluxInfer: Automatic diagnosis of performance anomaly for online database system. In: IPCCC. pp. 1–8.

Lou, J.-G., Lin, Q., Ding, R., Fu, Q., Zhang, D., Xie, T., 2013. Software analytics for incident management of online services: An experience report. In: ASE. pp. 475–485.

Lou, J.-G., Lin, Q., Ding, R., Fu, Q., Zhang, D., Xie, T., 2017. Experience report on applying software analytics in incident management of online service. Autom. Softw. Eng. 24 (4), 905–941.

Lu, X., Xie, Z., Li, Z., Li, M., Nie, X., Zhao, N., Yu, Q., Zhang, S., Sui, K., Zhu, L., Pei, D., 2022. Generic and robust performance diagnosis via causal inference for OLTP database systems. In: 2022 22th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. CCGRID.

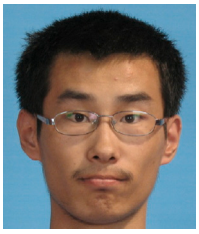
Ma, M., Lin, W., Pan, D., Wang, P., 2019. Ms-rank: Multi-metric and self-adaptive root cause diagnosis for microservice applications. In: ICWS. pp. 60–67.

Ma, M., Xu, J., Wang, Y., Chen, P., Zhang, Z., Wang, P., 2020. Automap: Diagnose your microservice-based web applications automatically. In: WWW. pp. 246–258.
MacDonell, S., Shepperd, M., Kitchenham, B., Mendes, E., 2010. How reliable are systematic reviews in empirical software engineering? IEEE Trans. Softw. Eng. 36 (5), 676–687.
Mahimkar, A.A., Ge, Z., Shaikh, A., Wang, J., Yates, J., Zhang, Y., Zhao, Q., 2009. Towards automated performance diagnosis in a large IPTV network. ACM SIGCOMM Comput. Commun. Rev. 39 (4), 231–242.
Maksai, A., Bogojeska, J., Wiesmann, D., 2014. Hierarchical incident ticket classification with minimal supervision. In: ICDM. pp. 923–928.
Malhotra, P., Vig, L., Shroff, G., Agarwal, P., 2015. Long short term memory networks for anomaly detection in time series. In: ESANN. Vol. 89, pp. 89–94.
Man, D., Yang, W., Wang, W., Xuan, S., 2012. An alert aggregation algorithm based on iterative self-organization. Procedia Eng. 29, 3033–3038.
Mani, S., Sankaranarayanan, K., Sinha, V.S., Devanbu, P., 2014. Panning requirement nuggets in stream of software maintenance tickets. In: FSE. pp. 678–688.
Mannila, H., Toivonen, H., 1996. Discovering generalized episodes using minimal occurrences. In: KDD. Vol. 96, pp. 146–151.
Marcu, P., Grabarnik, G., Luan, L., Rosu, D., Shwartz, L., Ward, C., 2009. Towards an optimized model of incident ticket correlation. In: IM. pp. 569–576.
Meng, F.J., Xu, J., Zhang, X., Yang, L., Chen, P., Wang, Y., Liu, X., Ayachitula, N., Murthy, K., Shwartz, L., et al., 2018. Opportunities and challenges towards cognitive IT service management in real world. In: SOSE. pp. 164–173.
Meng, Y., Zhang, S., Sun, Y., Zhang, R., Hu, Z., Zhang, Y., Jia, C., Wang, Z., Pei, D., 2020. Localizing failure root causes in a microservice through causality inference. In: IWQoS. pp. 1–10.
Miao, G., Moser, L.E., Yan, X., Tao, S., Chen, Y., Anerousis, N., 2010. Generative models for ticket resolution in expert networks. In: SIGKDD. pp. 733–742.
Miao, G., Tao, S., Cheng, W., Moulic, R., Moser, L.E., Lo, D., Yan, X., 2012. Understanding task-driven information flow in collaborative networks. In: WWW. pp. 849–858.
Mirheidari, S.A., Arshad, S., Jalili, R., 2013. Alert correlation algorithms: A survey and taxonomy. In: CSS. pp. 183–197.
Moral, C., de Antonio, A., Imbert, R., Ramírez, J., 2014. A survey of stemming algorithms in information retrieval. Inf. Res.: Int. Electron. J. 19 (1), n1.
Motahari-Nezhad, H.R., Bartolini, C., 2011. Next best step and expert recommendation for collaborative processes in it service management. In: BPM. pp. 50–61.
Nedelkoski, S., Cardoso, J., Kao, O., 2019. Anomaly detection and classification using distributed tracing and deep learning. In: CCGRID. pp. 241–250.
Nikzad-Khasmakhi, N., Balafar, M., Feizi-Derakhshi, M.R., 2019. The state-of-the-art in expert recommendation systems. Eng. Appl. Artif. Intell. 82, 126–147.
Niyazmand, T., Izadi, I., 2019. Pattern mining in alarm flood sequences using a modified PrefixSpan algorithm. ISA Trans. 90, 287–293.
Notaro, P., Cardoso, J., Gerndt, M., 2020. A systematic mapping study in AIOps. In: International Conference on Service-Oriented Computing. Springer, pp. 110–123.
Notaro, P., Cardoso, J., Gerndt, M., 2021. A survey of AIOps methods for failure management. ACM Trans. Intell. Syst. Technol. 12 (6), 1–45.
Palshikar, G.K., Vin, H.M., Saradhi, V.V., Mudassar, M., 2011. Discovering experts, experienced persons and specialists for it infrastructure support. Serv. Sci. 3 (1), 1–21.
Papagiannopoulou, E., Tsoumakas, G., 2020. A review of keyphrase extraction. Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 10 (2), e1339.
Porter, M.F., 1980. An algorithm for suffix stripping. Program.
Potgieter, B., Botha, J., Lew, C., 2005. Evidence that use of the ITIL framework is effective. In: NACCQ.
Potharaju, R., Jain, N., Nita-Rotaru, C., 2013. Juggling the jigsaw: Towards automated problem inference from network trouble tickets. In: NSDI. pp. 127–141.
Puech, T., Boussard, M., D'Amato, A., Millerand, G., 2019. A fully automated periodicity detection in time series. In: International Workshop on Advanced Analysis and Learning on Temporal Data. Springer, pp. 43–54.
Raimondi, F., Skene, J., Emmerich, W., 2008. Efficient online monitoring of web-service SLAs. In: FSE. pp. 170–180.
Ramaki, A.A., Khosravi-Farmad, M., Bafghi, A.G., 2015. Real time alert correlation and prediction using Bayesian networks. In: ISCISC. pp. 98–103.
Rance, S., 2011. Change management. ITIL® Serv. Transit. 60–89.
Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., Zhang, Q., 2019. Time-series anomaly detection service at microsoft. In: SIGKDD. pp. 3009–3017.
Reshef, D.N., Reshef, Y.A., Finucane, H.K., Grossman, S.R., McVean, G., Turnbaugh, P.J., Lander, E.S., Mitzenmacher, M., Sabeti, P.C., 2011. Detecting novel associations in large data sets. Science 334 (6062), 1518–1524.
Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "Why should i trust you?" Explaining the predictions of any classifier. In: SIGKDD. pp. 1135–1144.
Rijal, L., Colomo-Palacios, R., Sánchez-Gordón, M., 2022. Aiops: A multivocal literature review. Artif. Intell. Cloud Edge Comput. 31–50.
Sabharwal, N., Bhardwaj, G., 2022. What is AIOps? In: Hands-on AIOps. Springer, pp. 1–17.
Sadoddin, R., Ghorbani, A., 2006. Alert correlation survey: framework and techniques. In: Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services. pp. 1–10.
Salah, S., Maciá-Fernández, G., Díaz-Verdejo, J.E., 2013. A model-based survey of alert correlation techniques. Comput. Netw. 57 (5), 1289–1317.
Sang, B., Zhan, J., Lu, G., Wang, H., Xu, D., Wang, L., Zhang, Z., Jia, Z., 2011. Precise, scalable, and online request tracing for multitier services of black boxes. IEEE Trans. Parallel Distrib. Syst. 23 (6), 1159–1167.
Schad, J., Sambasivan, R., Woodward, C., 2022. Predicting help desk ticket reassignments with graph convolutional networks. Mach. Learn. Appl. 7, 100237.
Seth, A.K., 2010. A MATLAB toolbox for Granger causal connectivity analysis. J. Neurosci. Methods 186 (2), 262–273.
Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N., 2008a. Easyticket: A ticket routing recommendation engine for enterprise problem resolution. Proc. VLDB Endow. 1 (2), 1436–1439.
Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N., 2008b. Efficient ticket routing by resolution sequence mining. In: SIGKDD. pp. 605–613.
Shetty, M., Bansal, C., Kumar, S., Rao, N., Nagappan, N., 2022. SoftNER: Mining knowledge graphs from cloud incidents. Empir. Softw. Eng. 27 (4), 1–34.
Shetty, M., Bansal, C., Kumar, S., Rao, N., Nagappan, N., Zimmermann, T., 2021. Neural knowledge extraction from cloud service incidents. In: ICSE-SEIP. pp. 218–227.
Shimpi, V., Natu, M., Sadaphal, V., Kulkarni, V., 2014. Problem identification by mining trouble tickets. In: COMAD. pp. 76–86.
Siffer, A., Fouque, P.-A., Termier, A., Largouet, C., 2017. Anomaly detection in streams with extreme value theory. In: SIGKDD. pp. 1067–1075.
Sigelman, B.H., Barroso, L.A., Burrows, M., Stephenson, P., Plakal, M., Beaver, D., Jaspan, S., Shanbhag, C., 2010. Dapper, a large-scale distributed systems tracing infrastructure.
Silva, D., Dell'Amico, M., Hart, M., Roundy, K.A., Kats, D., 2018. Hierarchical incident clustering for security operation centers. In: IDEA.
Singh, J., Gupta, V., 2017. A systematic review of text stemming techniques. Artif. Intell. Rev. 48 (2), 157–217.
Soldani, J., Brogi, A., 2022. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. ACM Comput. Surv. 55 (3), 1–39.
Solé, M., Muntés-Mulero, V., Rana, A.I., Estrada, G., 2017. Survey on models and techniques for root-cause analysis. arXiv preprint arXiv:1701.08546.
Spirtes, P., Glymour, C.N., Scheines, R., Heckerman, D., 2000. Causation, Prediction, and Search. MIT Press.
Spitzer, F., 2013. Principles of Random Walk, Vol. 34, Springer Science & Business Media.
Srikant, R., Agrawal, R., 1996. Mining quantitative association rules in large relational tables. In: SIGMOD. pp. 1–12.
łgorzata Steinder, M., Sethi, A.S., 2004. A survey of fault localization techniques in computer networks. Sci. Comput. Program. 53 (2), 165–194.
Sun, H., Srivatsa, M., Tan, S., Li, Y., Kaplan, L.M., Tao, S., Yan, X., 2014. Analyzing expert behaviors in collaborative networks. In: SIGKDD. pp. 1486–1495.
Sun, P., Tao, S., Yan, X., Anerousis, N., Chen, Y., 2010. Content-aware resolution sequence mining for ticket routing. In: BPM. pp. 243–259.
Tang, L., Li, T., Pinel, F., Shwartz, L., Grabarnik, G., 2012. Optimizing system monitoring configurations for non-actionable alerts. In: NOMS. pp. 34–42.
Tang, L., Li, T., Shwartz, L., Grabarnik, G., 2013a. Recommending resolutions for problems identified by monitoring. In: IM. pp. 134–142.
Tang, L., Li, T., Shwartz, L., Grabarnik, G.Y., 2013b. Identifying missed monitoring alerts based on unstructured incident tickets. In: Proceedings of the 9th International Conference on Network and Service Management. CNSM 2013, IEEE, pp. 143–146.
Tang, L., Li, T., Shwartz, L., Pinel, F., Grabarnik, G.Y., 2013c. An integrated framework for optimizing automatic monitoring systems in large IT infrastructures. In: SIGKDD. pp. 1249–1257.
Thalheim, J., Rodrigues, A., Akkus, I.E., Bhatotia, P., Chen, R., Viswanath, B., Jiao, L., Fetzer, C., 2017. Sieve: Actionable insights from monitored metrics in distributed systems. In: Middleware. pp. 14–27.
Toutanova, K., Klein, D., Manning, C.D., Singer, Y., 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In: HLT-NAACL. pp. 252–259.
Toutanova, K., Manning, C., 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In: EMNLP/VLC.
Valdes, A., Skinner, K., 2001. Probabilistic alert correlation. In: International Workshop on Recent Advances in Intrusion Detection. pp. 54–68.
Wang, Y., Li, G., Wang, Z., Kang, Y., Zhou, Y., Zhang, H., Gao, F., Sun, J., Yang, L., Lee, P., et al., 2021. Fast outage analysis of large-scale production clouds with service correlation mining. In: ICSE. pp. 885–896.
Wang, P., Xu, J., Ma, M., Lin, W., Pan, D., Wang, Y., Chen, P., 2018a. Cloudranger: Root cause identification for cloud native systems. In: CCGRID. pp. 492–502.
Wang, J., Yang, F., Chen, T., Shah, S.L., 2015. An overview of industrial alarm systems: Main causes for alarm overloading, research status, and open problems. IEEE Trans. Autom. Sci. Eng. 13 (2), 1045–1061.
Wang, Q., Zeng, C., Iyengar, S., Li, T., Shwartz, L., Grabarnik, G.Y., 2018b. AISTAR: an intelligent system for online IT ticket automation recommendation. In: Big Data. pp. 1875–1884.
Wang, Q., Zhou, W., Zeng, C., Li, T., Shwartz, L., Grabarnik, G.Y., 2017. Constructing the knowledge base for cognitive it service management. In: SCC. pp. 410–417.
Wei, X., Sailer, A., Mahindru, R., Kar, G., 2007. Automatic structuring of it problem ticket data for enhanced problem resolution. In: IM. pp. 852–855.

Welch, T.A., 1984. Technique for high-performance data compression. Computer (52).
Weng, J., Wang, J.H., Yang, J., Yang, Y., 2018. Root cause analysis of anomalies of multitier services in public clouds. IEEE/ACM Trans. Netw. 26 (4), 1646–1659.
Wohlin, C., 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering. pp. 1–10.
Wu, L., Tordsson, J., Elmroth, E., Kao, O., 2020. Microrca: Root cause localization of performance issues in microservices. In: NOMS. pp. 1–9.
Wu, X., Turner, D., Chen, C.-C., Maltz, D.A., Yang, X., Yuan, L., Zhang, M., 2012. NetPilot: Automating datacenter network failure mitigation. In: SIGCOMM. pp. 419–430.
Xiao, Y., Watson, M., 2019. Guidance on conducting a systematic literature review. J. Plan. Educ. Res. 39 (1), 93–112.
Xu, J., He, R., 2018. Expert recommendation for trouble ticket routing. Data Knowl. Eng. 116, 205–218.
Xu, J., He, R., Zhou, W., Li, T., 2018a. Trouble ticket routing models and their applications. IEEE Trans. Netw. Serv. Manag. 15 (2), 530–543.
Xu, J., Mu, J., Chen, G., 2020. A multi-view similarity measure framework for trouble ticket mining. Data Knowl. Eng. 127, 101800.
Xu, J., Tang, L., Li, T., 2016. System situation ticket identification using SVMs ensemble. Expert Syst. Appl. 60, 130–140.
Xu, J., Wang, Y., Chen, P., Wang, P., 2017. Lightweight and adaptive service api performance monitoring in highly dynamic cloud environment. In: SCC. pp. 35–43.
Xu, Y., Wang, J., Yu, Y., 2019. Alarm event prediction from historical alarm flood sequences based on Bayesian estimators. IEEE Trans. Autom. Sci. Eng. 17 (2), 1070–1075.
Xu, J., Zhang, H., Zhou, W., He, R., Li, T., 2018b. Signature based trouble ticket classification. Future Gener. Comput. Syst. 78, 41–58.
Yang, L., Chen, J., Wang, Z., Wang, W., Jiang, J., Dong, X., Zhang, W., 2021. Semi-supervised log-based anomaly detection via probabilistic label estimation. In: ICSE. pp. 1448–1460.
Yang, F., Guo, C., 2017. Survey on advanced alarm strategies based on multivariate analysis. In: AdCONIP. pp. 612–617.
Yang, J., Li, L., Shen, S.-B., Yang, C.C.-Y., 2011. A QoS approach for detecting and managing a fault alarm storm. In: 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery. pp. 123–129.
Yin, Y., Yao, D., 2016. Causal inference based on the analysis of events of relations for non-stationary variables. Sci. Rep. 6 (1), 1–7.
Yu, B., 2008. An evaluation of text classification methods for literary study. Lit. Linguist. Comput. 23 (3), 327–343.
Yu, G., Chen, P., Chen, H., Guan, Z., Huang, Z., Jing, L., Weng, T., Sun, X., Li, X., 2021. MicroRank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In: WWW. pp. 3087–3098.
Yuan, S., Zhang, Y., Tang, J., Hall, W., Cabotà, J.B., 2020. Expert finding in community question answering: a review. Artif. Intell. Rev. 53 (2), 843–874.
Zang, T., Yun, X., Zhang, Y., 2008. A survey of alert fusion techniques for security incident. In: 2008 the Ninth International Conference on Web-Age Information Management. pp. 475–481.
Zeng, C., Li, T., Shwartz, L., Grabarnik, G.Y., 2014. Hierarchical multi-label classification over ticket data using contextual loss. In: NOMS. pp. 1–8.
Zeng, C., Zhou, W., Li, T., Shwartz, L., Grabarnik, G.Y., 2017. Knowledge guided hierarchical multi-label classification over ticket data. IEEE Trans. Netw. Serv. Manag. 14 (2), 246–260.
Zhang, Y., Guan, Z., Qian, H., Xu, L., Liu, H., Wen, Q., Sun, L., Jiang, J., Fan, L., Ke, M., 2021. CloudRCA: A root cause analysis framework for cloud computing platforms. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. Association for Computing Machinery, New York, NY, USA, URL: https://doi.org/10.1145/3459637.3481903.
Zhang, K., Kalander, M., Zhou, M., Zhang, X., Ye, J., 2020. An influence-based approach for root cause alarm discovery in telecom networks. In: ICSOC. pp. 124–136.
Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., et al., 2019. Robust log-based anomaly detection on unstable log data. In: ESEC/FSE. pp. 807–817.
Zhao, N., Chen, J., Peng, X., Wang, H., Wu, X., Zhang, Y., Chen, Z., Zheng, X., Nie, X., Wang, G., et al., 2020a. Understanding and handling alert storm for online service systems. In: ICSE-SEIP. pp. 162–171.
Zhao, N., Chen, J., Wang, Z., Peng, X., Wang, G., Wu, Y., Zhou, F., Feng, Z., Nie, X., Zhang, W., et al., 2020b. Real-time incident prediction for online service systems. In: ESEC/FSE. pp. 315–326.
Zhao, N., Jin, P., Wang, L., Yang, X., Liu, R., Zhang, W., Sui, K., Pei, D., 2020c. Automatically and adaptively identifying severe alerts for online service systems. In: INFOCOM. pp. 2420–2429.
Zhou, W., Li, T., Shwartz, L., Grabarnik, G.Y., 2015a. Recommending ticket resolution using feature adaptation. In: CNSM. pp. 15–21.
Zhou, W., Tang, L., Li, T., Shwartz, L., Grabarnik, G.Y., 2015b. Resolution recommendation for event tickets in service management. In: IM.
Zhou, W., Tang, L., Zeng, C., Li, T., Shwartz, L., Grabarnik, G.Y., 2016. Resolution recommendation for event tickets in service management. IEEE Trans. Netw. Serv. Manag. 13 (4), 954–967.
Zhou, W., Xue, W., Baral, R., Wang, Q., Zeng, C., Li, T., Xu, J., Liu, Z., Shwartz, L., Ya. Grabarnik, G., 2017. Star: A system for ticket analysis and resolution. In: SIGKDD. pp. 2181–2190.
Zimek, A., Schubert, E., Kriegel, H.-P., 2012. A survey on unsupervised outlier detection in high-dimensional numerical data. Stat. Anal. Data Min.: ASA Data Sci. J. 5 (5), 363–387.
Zong, B., Wu, Y., Song, J., Singh, A.K., Cam, H., Han, J., Yan, X., 2014. Towards scalable critical alert mining. In: SIGKDD. pp. 1057–1066.

Qingyang Yu received his bachelor's degree in Software Engineering from Shandong University in 2014. He received his master's degree in Computer Technology from the University of Chinese Academy of Sciences in 2017. He is currently a Ph.D. candidate in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests lie in Artificial Intelligence for IT Operations (AIOps), especially in trace anomaly detection and alert management.

Nengwen Zhao received her bachelor's degree in Computer Science and Technology from Wuhan University in 2017. She is currently a Ph.D. candidate in the Department of Computer Science, Tsinghua University, Beijing, China. Her research interests lie in AIOps, especially in alert management, log anomaly detection and change management.



Mingjie Li received his bachelor's degree in Computer Science and Technology from Tsinghua University in 2018. He is currently a Ph.D. candidate in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests include AIOps and Software Engineering.

Zeyan Li received his bachelor's degree in Computer Science and Technology from Tsinghua University in 2018. He is currently a Ph.D. candidate in the Department of Computer Science, Tsinghua University, Beijing, China. His research interests lie in AIOps, especially in root cause localization.

Honglin Wang received her bachelor's degree in Computer Science and Technology from Beijing University of Posts and Telecommunications in 2020. She is currently an algorithm engineer in Bizseer Technology, Beijing, China. Her research interests lie in AIOPs, especially in alert analysis.

Wenchi Zhang received his bachelor's degree in Computer Science from Northeastern University in 2018. He is currently an algorithm engineer in Bizseer Technology, Beijing, China. His research interests include time series anomaly detection, time series forecasting and alert analysis.

Kaixin Sui received her Ph.D. degree in Computer Science from Tsinghua University in 2016. She currently leads the Algorithm Research & Development Team in BizSeer Technologies Co., Ltd., Beijing, China. Her research interests lie in AIOps, including anomaly detection, root-cause analysis, system performance optimization and so on.

Dan Pei received the B.S. and M.S. degrees from Tsinghua University, Beijing, China, in 1997 and 2000, respectively, and the Ph.D. degree from the University of California, Los Angeles (UCLA), in 2005. He is currently an associate professor at Tsinghua University. His research interests are management and improvement of the performance and security of networked services. Right now he is focusing on the field of AIOps. He is a IEEE senior member.