# IT Incident Management by Analyzing Incident Relations

Rong Liu and Juhnyoung Lee

IBM Research
19 Skyline Drive, Hawthorne, NY 10532, USA
{rliu,jyl}@us.ibm.com

**Abstract.** IT incident management aims to maintain high levels of service quality and availability by restoring normal service operations as quickly as possible and minimizing business impact. Enterprises often maintain many applications to support their business. It is a significant challenge to diagnose incidents at application level due to complicated causes often aggregated from the shared IT environment, network, hardware, software, and changes. In this paper, we present a new approach to diagnosing application incidents by effectively searching for relevant co-occurring and reoccurring incidents. These relevant incidents reveal patterns of application failures and provide insights into incident resolution and prevention. This paper also provides a case study where we implement this approach and evaluate its performance in terms of search accuracy.

**Keywords.** Incident management, IT service management, incident relation, co-occurrence, reoccurrence, text analytics.

## 1 Introduction

The objective of the *IT Incident Management* is to restore normal service operations quickly to minimize business impact, thus ensuring high levels of service quality and availability [4]. An incident is any event which is not part of the standard operation of a service and which causes, or may cause, an interruption to or a reduction in the quality of that service. Incidents are the result of failures or errors in IT infrastructure. Incident management becomes more important as IT's contribution to business is ever growing. It also faces increasing challenges because an enterprise often maintains many applications in a shared IT environment composed of thousands of interdependent IT components, e.g. network, hardware, software etc. Incident diagnosis often requires investigation on complicated causes aggregated from this environment. Thus a sophisticated analytical platform is needed to aggregate events from multiple sources, detect suspected causes, suggest resolution, and predict potential failures.

In this paper, we present a new IT incident management approach to diagnosing incidents by effectively searching for relevant *co-occurring* and *reoccurring* incidents. Co-occurring incidents happen at different IT components concurrently and are possibly caused by the same root causes. Reoccurring incidents repeat over time with similar symptoms or features. These relevant incidents together can reveal patterns of application incidents, helping subject matter experts (SMEs) to reason about root

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 631–638, 2012.

causes of incidents and accelerate incident resolution. This paper also presents a case study where we implemented and tested this approach. Since our approach is developed based on a generic incident data structure, it can be applied in similar scenarios in incident management, for example, IT help desk support. The rest of this paper is structured as follows. Section 2 describes a motivating real-world case. In Section 3, we discuss the technical details of the proposed approach. Section 4 describes the implementation and reports evaluation results. Section 5 compares this approach with related work. Finally, Section 6 concludes this paper with future work outlined.

## 2 Case Study

A large corporation in the IT industry has over a thousand applications to support its business ranging from large-scale packaged applications (e.g. SAP) to small proprietary systems, which run in a shared, dynamic IT infrastructure. A critical objective is to maintain high-level application availability and reduce outages. It is challenging to diagnose incidents at the application level, because those incidents are often aggregated effects from problems in other lower layers. For example, the enterprise had a recent incident that blocked online orders for software. Meanwhile, another application supporting software downloading also failed. An exhaustive investigation led to a highly suspected cause that a dependent application for authenticating customers failed because of a storage area network outage happened in another geographic area.

Although this company has an integrated system for reporting and managing all IT incidents, relevant incidents could not be easily discovered for a few reasons. First, IT components are often managed by workgroups organized by expertise and by geography in a matrix structure. Due to this separation, relevant incidents may not be well communicated across workgroups. Second, the entire IT platform involves extreme complicated *dependencies* among IT components. Without deep knowledge about such dependencies, it would be impossible to scope relevant ones out of a huge number of incidents. Moreover, such dependencies are under constant evolution as the platform changes (e.g., provisioning new servers). Finally, useful information about an incident, such as affected IT components, symptoms, diagnosis results, is often recorded as *free-form text*. A typical incident is shown in Table 1. Another phenomenon is the frequent use of *ambiguous acronyms*. For example, depending on the context, "HRS" may mean "Hostname Resolution System" (an application name) or "Hours". Searching incidents only by keywords without considering their context is often deficient. Next, we propose a new search method to overcome these challenges.

## 3 Search for Relevant Incidents

As illustrated by the case study, an incident is often not an isolated event. It can be diagnosed by finding relevant co-occurring and reoccurring incidents and consolidating them to discover insights regarding how it happened and how it can be fixed. Our search algorithm consists of three steps: classifying incidents, searching incident by keywords, and calculating relevancy score and ranking search results.

### 3.1 Incident Classification

A common practice in IT Incident Management is to classify incidents by proper categories [4,8]. For example, we classify the incidents in the case study by keywords from these facets [6]: application, server, middleware, infrastructure, and symptom. It is often feasible to obtain decent vocabulary for these facets. For instance, companies usually maintain lists of applications, servers and middleware as part of their asset portfolio. Such lists become good dictionaries. One can also gather a list of frequently used terms as a dictionary, for instance, for symptom facet. With these dictionaries, keywords can be extracted from incident text by using text analysis software, for instance, IBM Context Analytics (ICA) [3]. Synonyms and annotation patterns are used to improve the accuracy of extraction. For example, MQ is a synonym of MQSeries. After classification, an incident can be represented by a bag of keywords. For instance, incident IN1 in Table 1 is classified by keywords as shown in Figure 1. These keywords are referred to as *classification keywords* in this paper.

**Table 1.** An Example of Application Incidents

| Attributes | Value |
|---|---|
| Incident ID | IN1 |
| Problem Abstract | ServerXYZ Issue: MQ connectivity has been reported lost. |
| Problem Description | Ticket#: Sev1; Application: SomeApp; Server/URL: ServerXYZ Issue: MQ connectivity has been reported lost. ITD (InternetService) team should verify and "run mustGather and recycle SomeApp (on ServerXYZ and ServerXYZ 2 as needed; Duty manager needed or not (not at this time); if so, impact: No revenue impact. Business impact is to some of the SomeApp (software service) orders (but not all); Team to be engaged: InternetService |
| Problem Result | SomeApp recycled |
| Occurred Time | 2011-12-25 08:28:00 |
| Solved Time | 2011-12-25 09:25:40 |
| Account ID | SomeAccount |
| Resolver Group | SomeGroup |

After classification, a critical step is to validate the accuracy of each extracted keyword and assign an appropriate accuracy weight that can be used for discovering relevant incidents later. This validation is to ensure (1) an acronym is semantically correct, and (2) the combination of extracted keywords for an incident is valid against proper domain knowledge. For example, we need to check if acronym “HRS” indeed means “Hostname Resolution System”. We first check if its full name can be found in the incident text. If it cannot be found, we rely on other extracted keywords or attributes to infer the meaning as in (2). For the example shown in Figure 2, we have learned that “SomeApp” is hosted in server “ServerXYZ” and requires middleware “MQSeries”. With this application architecture information, we can confirm that the combination of keywords {SomeApp, ServerXYZ, mqseries} is correct. In case domain knowledge is not available, we can learn it dynamically by checking co-occurrence of keywords [1]. For example, if the joint probability of keyword

"SomeApp" and "ServerXYZ2" is higher than a certain level, we can infer that "ServerXYZ2" may be a hosting server of "SomeApp". After validation, we assign an accuracy weight $w_1$ to each extracted keyword to indicate the level of confidence on its validity.

| | |
|---|---|
| Application: | *SomeApp* |
| Server: | *serverXYZ, ServerXYZ2* |
| Middleware: | *mqseries* |
| Symptom: | *mqseries connectivity* |

**Fig. 1.** Classification of Example 1

### 3.2 Relevant Incident Search

After classification, we structure incident information out of the free-form text. Then we design a hybrid search engine integrating both faceted search and free text search to discover *co-occurring* and *re-occurring* incidents. To simplify its use, the search engine requires only an incident ID as an input, and automatically decides appropriate *search keywords* and ranks returned results by relevancy from high to low.

To find reoccurring incidents, the search engine uses classification keywords as search terms. For example, Figure 1 shows all search keywords for finding reoccurring incidents of IN1 (see Table 1). It takes additional consideration to find co-occurring incidents. First, co-occurring incidents happen about the same time. Hence, a mandatory time constraint is placed to limit the search scope to recent incidents. Second, co-occurring incidents may be reoccurring events. Therefore, classification keywords are also taken. Moreover, co-occurring incidents may indicate dependencies among involved IT components. Hence, we also add keywords representing dependent components as search terms. We refer to these keywords as *dependency keywords*. Taking the same example of IN1, to find co-occurring incidents, two dependent applications, "DepApp1" and "DepApp2", are added. A boosting weight ($w_2$) can be assigned to each search keyword based on its impact. Our search engine assigns a default weight for each dependency keyword, for instance, $w_2 = 2$.

With search keywords defined, incidents that satisfy any of the search keywords and mandatory time constraints are returned to achieve a high recall rate. For instance, to search for co-occurring incidents for IN1, its search keywords and time constraints are represented as a query shown in Figure 2. This query consists of both structured and unstructured portions. The structured query searches for incidents by using the incident classification keywords and structured fields in database tables. The unstructured part handles the need for free text search, for example, incident symptoms. The final search result is the union of those returned from the two queries.

Structured:
(*application* in ("*SomeApp*", "*DepApp1*", "*DepApp2*") or *server* in ("*ServerXYZ*", "*ServerXYZ2*") or *middleware* in ("*mqseries*")) and *occurred_time* between ("*12/22/2011*", "*12/25/2011*")

Unstructured:
+mq +connectivity date>="2011-12-22" date<="2011-12-25"

**Fig. 2.** Formulate Search Keywords as Queries

With a large number of results returned from search, we adapt vector-space model [7] slightly to calculate the similarity score for returned incidents. Based on this model, a document is represented as a vector of keywords $v = (x_1, x_2, \ldots, x_n)$ in a $n$-dimensional vector space, where $x_i$ is the weight of keyword $i$. $|v|^2 = \sum_n x_i^2$ is the length of the vector. The similarity between $(v_1, v_2)$ is the cosine of the angle $\theta$ between them, i.e., $s = \cos(\theta) = \frac{v_1 \cdot v_2}{|v_1|\,|v_2|}$, where $v_1 \cdot v_2 = \sum_n x_{1i} x_{2i}$. The vector space in our case contains the classification keywords of incidents. The vector of an incident contains all of its classification keywords, and conditionally the dependency keywords. When a dependency keyword is found in a returned incident, it is added to the vector to boost the similarity score. The weight of a keyword is $x = w_1 w_2$, where $w_1$ is the accuracy weight and $w_2$ is the boosting weight. Note that in a regular vector-space model, keyword frequency is often an important factor for weight. However, here incident text is often dominated by technique specification or message logs. For instance, a server name appears many times in a log. Frequency-based weights may favor incidents with lengthy messages and have negative impact on search precision.

To illustrate, consider two relevant incidents IN2 ($v_2$) and IN3 ($v_3$) for incident IN1 ($v_1$) as shown in Table 2. $v_1$ has five classification keywords ($w_1$=1) and two dependency keywords ($w_2$ = 2). $v_2$ matches three of the classification keyword (i.e., $v_1 \cdot v_2 = 3$). The similarity score between $v_1$ and $v_2$ is $\frac{3}{\sqrt{5}*\sqrt{3}} = 0.77$. $v_3$ matches one classification keyword and one dependency keyword. Thus, $v_1 \cdot v_3$= 1*1+2*2=5 and the length of $v_1$ is $5 + 2^2 = 9$. The similarity score between $v_1$ and $v_3$ is $\frac{5}{\sqrt{9}*\sqrt{5}} = 0.75$.

The similarity score considers whether two incidents are similar to each other in terms of classification keywords, but it may not be sufficient for finding truly relevant incidents. Take a query with two keywords {"db2", "SomeApp"} as an example. This query may return a large number of incidents because db2 is a widely used component. Among them, many incidents are about general DB2 issues irrelevant to specific applications. However, since these incidents are classified by only "DB2" keyword, their similarity score $\frac{1}{\sqrt{2}*1} = 71\%$ is actually pretty high.

**Table 2.** Calculating Similarity Score

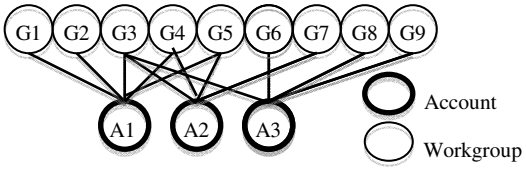
| Incident | | Keyword Vector | | | | | | | $v_1 \cdot v_2$ (or $v_3$) | $\lvert v_1\rvert*\lvert v_2\rvert$ (or $\lvert v_3\rvert$) | *Similarity* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *SomeApp* | *ServerXYZ* | *ServerXYZ2* | *mq* | *mq conn.* | *dependent app* | | | | |
| | | | | | | | *DepApp1* | *DepApp2* | | | |
| $v_1$ | IN1 | 1 | 1 | 1 | 1 | 1 | | 2 | | | |
| $v_2$ | IN2 | 1 | 1 | | 1 | | | | 3 | $\sqrt{5} * \sqrt{3}$ | 0.77 |
| $v_3$ | IN3 | | 1 | | | | | 2 | 5 | $\sqrt{9} * \sqrt{5}$ | 0.75 |

In order to filter irrelevant results, we use attributes other than the classification keywords to infer a broad context. In the IT incident management domain, an account is a well-accepted concept representing an organization unit responsible for the resolution of incidents in a particular business area. In general, each account involves general-purpose support workgroups, such as Network Support team, and specialized support team, e.g. Internet Service support team. The technology configuration of an account thus can be inferred from the specialized workgroups. We plot a diagram

among accounts and specialized workgroups, where each node is either a workgroup or an account and the link between an account and a workgroup indicates the workgroup is called for the account's incidents. An example is shown in Figure 3(a). This diagram illustrates the technology configuration and similarity among accounts by technology configuration. For example, account A1 engages workgroup G1-5, while A2 involves workgroups G3-5 and G7. Using the same vector-space model, the similarity between A1 and A2 is estimated to be 67%, as shown in Figure 3(b). We use this similarity to estimate context relevancy ($w_3$) because shared workgroups indicate technology dependencies and compatibility between each other. For each incident returned from a search, the final relevancy score $r = f(s, w_3)$. For instance, a simple function we use is $r = \alpha * w_3 + (1- \alpha) * s$, where $0< \alpha<1$. Given that three incidents in Table 2 belong to account A1, A2, and A3 respectively, with $\alpha=0.5$, we can get the final relevancy score for $v_2$ and $v_3$ are 0.72 and 0.48 respectively.

### *3.3* Root Cause Analysis Using Relevant Incidents

Continuous analysis on a group of relevant incidents often provides insights for SMEs to diagnose root causes and define proactive actions to prevent similar incidents. First, we can summarize commonalities of relevant incidents. These commonalities allow SMEs to pinpoint exact problems and exclude other suspected causes. For example, incident IN1 and its reoccurring incidents share common keywords: "ServerXYZ" and "mqseries", suggesting SMEs to focus on MQSeries installed on ServerXYZ server. As another example, if a group of re-occurring incidents all depend on an infrastructure component, this component is a highly suspected cause. Also, extracting common keywords from the resolution description of a group of reoccurring incidents can provide useful insights into resolution strategies. Second, a group of co-occurring incidents may disclose a causal event chain. Chaining these incidents along their occurrence times may disclose potential application dependencies. More advanced analysis can be conducted to correlate a group of relevant incidents with other data, for example, maintenance schedule, resource usage, and change events (i.e., application fix, upgrade). With the correlation, prediction rules can be configured to predict potential problems. For example, incident IN1 has been diagnosed in this way to find its root cause. A finding is that IN1 and its reoccurring incidents are highly correlated with messages alerting lost MQSeries connections after a server rebooted for regular maintenance. With this discovery, a rule is configured to alert SMEs when a server maintenance event is scheduled.




(a)

| | A1 | A2 | A3 |
|---|---|---|---|
| A1 | | 67% | 20% |
| A2 | 67% | | 45% |
| A3 | 20% | 45% | |

(b)

**Fig. 3.** Account Configuration & Similarity

## 4 Search Engine Implementation and Evaluation

Figure 4 shows the implemented architecture of our search engine. This engine has two types of flows: front-end (solid line) and backend (dotted line). At the backend, incidents data are extracted and fed into the IBM Context Analytics (ICA). Classification keywords are extracted, validated, and stored in a database as consolidated incident data. At the front-end, a user issues a query to the search engine and the search engine transforms this query into two sub-queries, one for structured search to the database and the other to ICA for free text search. Search results are consolidated and ranked. Since a large portion of the query is handled as a database query, the search is efficient. Our pilot test shows on average it takes less than a minute to complete a search for co-occurring/reoccurring incidents from about 140,000 incident records.

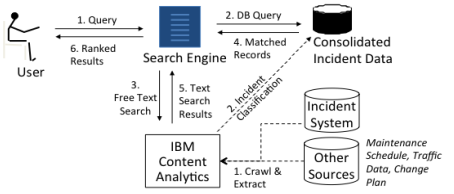



**Fig. 4.** System Architecture

**Table 3.** Evaluation Result

| Testing Cases | Result | | | |
|---|---|---|---|---|
| | *Success* | *Partial Success* | *Failure* | *Unknown* |
| Reoccurring | 15 | 1 | | |
| Non-reoccurring | 3 | 1 | 1 | 2 |
| *Total* | 18 | 2 | 1 | 2 |

This new search engine is evaluated by SMEs in a pilot test. SMEs carefully chose 23 incidents as testing cases. We designed a new evaluation metric combining both recall and precision rates. For each testing case, at maximum 20 relevant incidents are returned. SMEs rated each testing case: success, partial success, failure, or unknown. A testing case was rated success only if all SME-recognized relevant incidents (usually less than 20) were returned. A result was rated as "unknown" if it cannot be confirmed. Table 4 summarizes the result. The testing result is reasonably positive (78% success rate). Those testing cases rated "partial success" and "failure" are primarily caused by ambiguous acronyms or insufficient information disclosed in incidents.

## 5 Related Work

In the area of IT Service Management (ITSM), Information Technology Infrastructure Library (ITIL) provides a set of practices that focuses on aligning IT services with the needs of business [4,8]. The work presented in this paper concerns mostly about the investigation and diagnosis of incidents in ITSM using text mining and statistical methods. Recently, there is growing interest in using statistical analytics to analyzing and managing incidents. In [9], an ensemble of Tree-Augmented Bayesian Network models is provided to correlate workload metrics with service level objectives. A service Delivery Portal introduced in [5] provides a set of technologies to help system administrators (SA) to diagnose and manage incidents. This platform aggregates various incident data sources and allows SA to search for relevant events based on keywords or incident attributes. Our approach improves the search accuracy by considering domain knowledge and incident context during search. Another broad

domain related to this work is text mining [1,6] and machine learning [2]. A comprehensive survey of techniques in this domain is provided by [1]. In our work, we adapted Vector Space Model to calculate relevancy score between incidents. Another technique used by our work is co-occurrence networks, which represent the collective interconnection of terms based on their paired presence within a specified unit of text [1]. We apply this concept to automatically learn keyword dependency as domain knowledge.

## 6 Concluding Remarks and Future Work

IT incident management, which ensure high levels of service quality and availability, is a significant challenge primarily because enterprises often maintain many applications in shared dynamic IT environments. In this paper, we present a new approach to diagnosing application incidents by effectively searching for relevant co-occurring and reoccurring incidents. We designed a hybrid search engine that finds relevant incidents in both structured and unstructured formats. These relevant incidents together reveal underlying patterns of incidents and then provide SMEs insights into incident causes and resolution. We implemented this approach and evaluated its performance in terms of search accuracy. The pilot test shows that this approach is reasonably effective in discovering relevant incidents. Our future work is to develop predictive modeling capability based on relevant incidents discovered. We also plan to enhance its root cause analysis capability with a richer set of test data and test cases.

## References

1. Berry, M.W., Castellanos, M.: Survey of Text Mining I: Clustering, Classification, and Retrieval, 2nd edn. Springer (2007)
2. Bishop, C.: Pattern Recognition and Machine Learning. Springer (2006)
3. IBM Content Analytics with Enterprise Search, http://www-01.ibm.com/software/ecm/content-analytics/bundle.html
4. ITIL Incident Management - The ITIL Open Guide, http://www.itlibrary.org/index.php?page=Incident_Management
5. Lenchner, J., Rosu, D., Velasquez, N.F., Guo, S., Christiance, K., DeFelice, D., Deshpande, P.M., Kummamuru, K., Kraus, N., Luan, L.Z., Majumdar, D., McLaughlin, M., Ofek-Koifman, S., Deepak, P., Perng, C.-S., Roitman, H., Ward, C., Young, J.: A service delivery platform for server management services. IBM Journal of Research and Development 53(6), 792–808 (2009)
6. Rodriguez-Castro, B., Glaser, H., Carr, L.: How to *Reuse* a Faceted Classification and Put It on the *Semantic* Web. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 663–678. Springer, Heidelberg (2010)
7. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM 18(11), 613–620 (1975)
8. Van Bon, J., Verheijen, T.: Frameworks for IT Management. Van Haren Publishing (2006) ISBN 9789077212905
9. Zhang, S., Cohen, I., Goldszmidt, M., Symons, J., Fox, A.: Ensembles of models for automated diagnosis of system performance problems. In: DSN 2005, Yokohama, Japan (2005)