Healing Online Service Systems via Mining Historical Issue Repositories

Rui Ding, Qiang Fu, Jian-Guang Lou, Qingwei Lin, Dongmei Zhang
Microsoft Research Asia, China
{juding, qifu, jlou, qlin, dongmeiz}@microsoft.com

Jiajun Shen
Shanghai Jiao Tong University, China
shenjiajun90@gmail.com

Tao Xie
North Carolina State University, USA
xie@csc.ncsu.edu

ABSTRACT

Online service systems have been increasingly popular and important nowadays, with an increasing demand on the availability of services provided by these systems, while significant efforts have been made to strive for keeping services up continuously. To assure the user-perceived availability of a service, reducing the Mean Time To Restore (MTTR) of the service remains a very important step. To reduce the MTTR, a common practice is to restore the service by identifying and applying an appropriate healing action (i.e., a temporary workaround action such as rebooting a SQL machine). However, manually identifying an appropriate healing action for a given new issue (such as service down) is typically time consuming and error prone. To address this challenge, in this paper, we present an automated mining-based approach for suggesting an appropriate healing action for a given new issue. Our approach generates signatures of an issue from its corresponding transaction logs and then retrieves historical issues from a historical issue repository. Finally, our approach suggests an appropriate healing action by adapting healing actions for the retrieved historical issues. We have implemented a healing-suggestion system for our approach and applied it to a real-world online service system that serves millions of online customers globally. The studies on 77 incidents (severe issues) over three months showed that our approach can effectively provide appropriate healing actions to reduce the MTTR of the service.

Categories and Subject Descriptors: D.2.5 [Software Engineering]: Testing and Debugging
General Terms: Management, Performance, Reliability
Keywords: Online service system, healing action

1. INTRODUCTION

Online service systems such as online banking, e-commerce, and email services have been increasingly popular and important nowadays, with an increasing demand on the availability of services provided by these systems. While significant efforts have been made to strive for keeping services up continuously, studies [6] on a sample of Internet hosts have shown that daily and weekly service downs still appeared very commonly in online services. A serious service down for a non-trivial period can result in huge economic loss or other serious consequences. For example, customers of a service provider may turn to competing providers if the offered services are not available for a non-trivial period every now and then.

To assure the user-perceived availability of a service [8], reducing the Mean Time To Restore (MTTR) of the service remains a very important step. In order to reduce the MTTR, a common practice is to restore the service by identifying and applying an appropriate healing action [2] (i.e., a temporary workaround action such as rebooting a SQL machine) after the occurrence of an issue (e.g., an unplanned interruption or degradation in the quality of an online service). Then after service restoration, identifying and fixing underlying root causes for the issue can be conducted via offline postmortem analysis. In this way, online application of an appropriate healing action for the issue wins time for offline diagnosis and fixing of underlying root causes (which typically take relatively longer time to resolve).

However, manually identifying an appropriate healing action for a given new issue is time consuming and error prone. Such manual process is typically based on investigating service-instrumented data such as transaction logs. According to an internal study from an online-service team, about 90% time of the MTTR is spent on manual effort for identifying an appropriate healing action. Such substantial manual effort is due to two factors. First, investigating a large amount of service-instrumented data is time consuming and requires domain knowledge. Second, searching for an appropriate healing action is laborious and inaccurate.

To address high cost and error proneness of manually identifying an appropriate healing action, in this paper, we present an automated mining-based approach for suggesting an appropriate healing action for a given new issue. Our approach generates signatures of an issue from its corresponding transaction logs and then retrieves historical issues (with signatures similar to the signatures of the issue) from a historical issue repository. Each historical issue in the repository contains its appropriate healing action taken by operators to heal the issue. Finally, our approach suggests an appropriate healing action by adapting healing actions for the retrieved historical issues.

In particular, our approach measures the similarity between the given new issue and a historical issue based on their characteristics. To characterize an issue, we extract features from the transaction logs for an issue. However, such characterization faces two major challenges due to the *high-correlation* phenomenon and *weak-discrimination* phenomenon (see detailed illustration in Section 2). In order to tackle these challenges, we develop the technique of concept analysis to address the high-correlation phenomenon and the technique of contrast analysis to address the weak-discrimination phenomenon.

We have implemented our approach as a healing system at the Software Analytics group of Microsoft Research Asia in collaboration with a Microsoft product team for online services. We have deployed our healing system on one online service (serving mil-

ASE'12, September 03–07, 2012, Essen, Germany.

lions of online customers globally) for more than three months. During this period, 76 operators in this product team effectively diagnosed and healed this online service with intensive assistance of our healing system. Our experiments on this real-world online service demonstrate the effectiveness of our approach.

In summary, this paper makes the following main contributions:

- We propose an automated mining-based approach that suggests an appropriate healing action for a given new issue by retrieving similar historical issues from a historical issue repository and adapting healing actions of these similar historical issues.
- We define a novel similarity metric for measuring similarity between the given new issue and historical issues. Based on the similarity metric, our techniques of concept analysis and contrast analysis help address challenges and achieve high accuracy on historical-issue retrieval.
- We implement and apply our approach on a real-world online service (serving millions of online customers globally). Our experiments on 77 incidents (severe issues) over three months show that our approach can effectively suggest appropriate healing actions to reduce the MTTR of the service.

The paper is organized as follows. Section 2 gives an illustrative example. Section 3 presents our approach. Section 4 presents experimental results. Section 5 discusses related work.

2. AN ILLUSTRATIVE EXAMPLE

Figure 1 shows a log stream collected for two example transaction instances (within the occurring period of an issue): the user-login transaction instance (the highlighted log entries sharing the same transaction ID) and file-editing transaction instance (the un-highlighted log entries sharing the same transaction ID). Such logs record relatively detailed information about run-time behaviors of a system. The event ID of a log entry can help map the log entry to a unique source-code location. Figure 2 illustrates the corresponding portion of source code for event ID (in short as event throughout the rest of the paper) "y1", which describes that a SQL exception has been thrown. Figure 3 shows log statistics for three transaction types within the occurring period of an issue, including the two transaction types of the two example transaction instances shown in Figure 1 in addition to the file-reading transaction type. The *http-status code* listed in the last two columns of Figure 3 indicates the returned status of a given transaction instance: "200" denotes "OK" and "500" denotes "Internal Server Error".

The *high-correlation* phenomenon refers to the correlation of events' occurrences. For example, events "x1"~"x8" always appear together to indicate invalid cookies. If we do not group them together when comparing event sequences for the given issue and historical issues, events "x1"~"x8" would contribute eight times than event "y1" to characterize the given new issue, likely causing this given issue to be wrongly matched with a historical issue with the dominating symposium as events "x1"~"x8". Such wrong matching causes a wrong healing action to be suggested.

The *weak-discrimination* phenomenon refers to noisy events that appear relatively independent to the transaction status (e.g., the http status). Log messages corresponding to these events contribute little to distinguish different types of issues. Thus, due to such *weakly-discriminating* events, retrieved historical issues for the given new issue may not be desirable. For example, assume that another different issue from the historical issue repository is dominated by events "a", "b", "d", "y2", and "z" where "y2" is related to "antivirus timeout". If we do not address such phenomenon, we would wrongly retrieve this historical issue for the given issue related to a SQL exception (since the only difference of events for these two issues is "y1" vs. "y2").

time	event	transaction ID	message
02/08/2012.	a	9d959c…	Request # entering …
02/08/2012.	b	9d959c…	created cookie handler with…
02/08/2012.	a	7d467b…	Request $ entering …
02/08/2012.	c	9d959c…	cookie with name '*' was read …
02/08/2012.	x1	7d467b…	SQL server * failover detected…
02/08/2012.	x1	7d467b…	SQL server * failover detected…
02/08/2012.	x2	7d467b…	$ is not sign…
02/08/2012.	x3	7d467b…	Building authentication url
02/08/2012.	d	9d959c…	Site=/*/*/…
02/08/2012.	x4	7d467b…	attempt to create a sign…
02/08/2012.	e	9d959c…	Detected use of * from …
02/08/2012.	b	7d467b…	created cookie handler with…
02/08/2012.	x5	7d467b…	writing cookie of $
02/08/2012.	x6	7d467b…	cookie of $ was not present.
02/08/2012.	y1	9d959c…	SqlException: server * was not …
02/08/2012.	z	9d959c…	# leaving monitored scope of …
02/08/2012.	x7	7d467b…	$ does not require ssl.
02/08/2012.	x8	7d467b…	redirecting $ to …
02/08/2012.	x2	7d467b…	$ is not sign…
02/08/2012.	z	7d467b…	$ leaving monitored scope of …

Figure 1. Log stream for example transaction instances within the occurring period of an issue

```
ULS.Logging(
 0x12f3ce22 /* y1*/, Database, "{0}", String.Format(CultureInfo.InvariantCulture,
"SqlException: '{0}' was not found. Source: '{1}' Procedure: '{2}', LineNumber: '{3}' …"));
```

Figure 2. Example logging statement in source code

transaction types	sequence of event	# transactions	
		status=500	*status=200*
file editing	a, b, c, d, e, y1, z	187	0
user login	a, x1, x1, x2, x3, x4, b, x5, x6, x7, x8, x2, z	36	0
file reading	a, b, c, d, e, z	0	13485

Figure 3. Log statistics for three transaction types within the occurring period of an issue

3. APPROACH

Our approach consists of three steps. First, we use concept analysis and contrast analysis to generate signatures for an issue. Second, we retrieve historical issues similar to the given new issue from the historical issue repository based on their generated signatures. Third, we produce healing suggestions by adapting the healing actions of the retrieved historical issues.

3.1 Signature Generation

Our approach includes the techniques of concept analysis and contrast analysis to address *high-correlation* and *weak-discrimination* phenomena.

3.1.1 Concept Analysis

In our problem, each transaction instance corresponds to an event sequence. However, we drop the information on the temporal order and event-recurrence count, and use an event set to correspond to each transaction instance. Then, we group highly-correlated events together by applying Formal Concept Analysis (FCA). The intuition is that highly-correlated events together indicate one kind of symptom.

Here we construct the concept lattice by applying FCA, and denote $Int(concept\_k)$ as the *intent* of the *k*-th concept.

3.1.2 Contrast Analysis

By inspecting all the edges in the concept lattice, our contrast analysis finds the complementary sets ($Int(parent)\backslash Int(child)$) that are highly correlated to failed transaction instances.

We next give a precise definition about the fail/success label for a transaction instance.

Fail/success label. We define the label for each transaction instance (reflecting the transaction status) as

$$label_i = \begin{cases} failure, \ if\ (HttpStatus_i \geq 500\ or\ Duration_i \geq 10s) \\ success, \qquad\qquad otherwise \end{cases}$$

where i denotes the index of a specific transaction instance.

Positive correlation. We calculate mutual information to measure the correlation between a concept and a failure. Let *x* and *y* be the number of failed and succeeded transaction instances for a given concept, respectively. Let *n* and *m* be the total number of failed and succeeded transaction instances within the occurring period of a given new issue, respectively. We treat these observations as a result of statistical sampling, and then define random variable X_c, Y as

$$X_c = \begin{pmatrix} 1 & if\ thsampled\ transaction\ instance\ belongs\ to\ concept\ c \\ 0 & otherwise \end{pmatrix}$$

$$Y = \begin{pmatrix} 1 & if\ sampled\ transaction\ instance\ fails \\ 0 & otherwise \end{pmatrix}$$

Then *x*, *y*, *n*, *m* are the observation that can reveal the joint distribution of X_c, *Y*.

We define the formula as below

$$M(X_c, Y) = P(X_c = 1, Y = 1) log \frac{P(X_c = 1, Y = 1)}{P(X_c = 1)P(Y = 1)} + P(X_c = 0, Y = 0) log \frac{P(X_c = 0, Y = 0)}{P(X_c = 0)P(Y = 0)}$$

Here we use only the positive correlation part of mutual information. In general, the negative correlation happens trivially often in network traces, and is not meaningful [5].

Delta information. To achieve accurate retrieval, we need to exclude noisy events and keep only "clean" events. We analyze *Delta Mutual Information* (*DMI*) between child and parent concept nodes in the concept-lattice graph, to measure how the delta events contribute to correlation.

We define

$$DMI(\Delta Es) = M(X_{Int(chi)}, Y) - M(X_{Int(par)}, Y)$$

$DMI(\Delta Es)$ represents the contribution of the extra events ΔEs for failure correlation.

Signature. By walking through each edge in the concept-lattice graph, we select ΔEs as a signature if it satisfies *criteriaX*:

$$criteriaX := \begin{cases} M(X_{Int(par)}, Y) > 0 \\ DMI(\Delta Es) > 0 \end{cases}$$

Next, we use the $DMI(\Delta Es)$ as the weight for signature ΔEs. So a signature $s_{\mathcal{F}}$ can be represented as follows:

$$s_{\mathcal{F}} = < \Delta Es, DMI(\Delta Es) > \ s.t.\ criteriaX\ are\ \text{satisfied.}$$

3.2 Similar-Issue Retrieval

In similar-issue retrieval, we first need to have a representation for each issue and then define an efficient similarity metric to measure similarity between two issues.

We treat each issue in the historical issue repository as one document, each signature as one term, and the corresponding DMI as the weight of each term.

So we represent each issue $d_i = \sum_{p \in A(i)} w_{ip} \overrightarrow{t_p}$, where a signature is represented by an abstract vector $\overrightarrow{t_p}$, p is the index of the signature in issue d_i, $A(i)$ is the index set, and w_{ip} is the *DMI* of signature $\overrightarrow{t_p}$ (here we use the *DMI* as the weight).

We calculate the cosine score of two document vectors as the similarity metric:

$$sim(d_i, d_j) = \frac{d_i \cdot d_j}{\| d_i \| \| d_j \|} = \frac{\sum_{p \in A(i), q \in A(j)} w_{ip} w_{jq} \overrightarrow{t_p} \cdot \overrightarrow{t_q}}{\| d_i \| \| d_j \|}$$

The metric measures the cosine of the angle between the two vectors. Here $\| d_i \| = \sqrt{d_i \cdot d_i}$

We define inner product between two terms $\overrightarrow{t_p} \cdot \overrightarrow{t_q}$:

$$\overrightarrow{t_p} \cdot \overrightarrow{t_q} = the\ number\ of\ overlapping\ events\ of\ p, q$$

3.3 Healing-Suggestion Adaptation

We use a triple structure $h_{\mathcal{F}} < verb, target, locaton >$ to represent a healing action.

Table 1. Combinations of healing actions

verb	***target***	**event of *location***		***verb***	***target***	**event of *location***
recycle	App-pool	ev1		re-image	WFE	ev1
restart	IIS	ev1		rotate	WFE	ev1
reboot	WFE	ev1		rotate	SQL	ev2
reboot	SQL	ev2		patch	WFE	ev1
reset	DB	ev2		patch	SQL	ev2

Based on empirical investigations of healing actions for online service systems, we find that most healing actions can be formatted as

Healing action = verb + target + location

When we retrieve a similar historical issue for the given new issue, we extract the *verb* and *target* from the description text of the historical issue. The *location* is the name of the exact affected machine with its physical location of the given new issue. We extract the *location* with a rule-based technique, which parses the log messages of a specific event in the logs of the given new issue. For example, when an issue related to a SQL exception occurred, the system produced many failed transaction logs, and such transaction logs contain events of a specific type (denoted as ev2), such as "connect to SQL35-003 timeout". From the events, we extract "SQL35-003" as the specific machine in a bad state. Similarly, some other events of another specific type (denoted as ev1) contain the name of the Frontend machine in a bad state. Events of these two types (ev1 or ev2) are identified according to the *verb* and *target* extracted from the retrieved historical issue. The total combinations (based on our current study) among the *verb*, *target* and *location* are illustrated in Table 1.

4. PRELIMINARY EXPERIMENTS

In our experiments, we intend to answer one major research question: how much effectively can our approach suggest appropriate healing actions for the given new issues?

4.1 Experiment Setup

We have deployed our healing system in a released production service, named ServiceX (instead of the real name due to confidentiality). We tracked issues for more than three months, from

Nov. 1, 2011 to Feb. 18, 2012. There were in total 332 resolved issues where 146 issues have clear description information. Among the 146 issues, 69 issues are trivial to our experiments such as service upgrade or are still under investigation with unclear healing actions, and these 69 issues are excluded from our experiments. The healing actions for the remaining 77 issues are categorized into 8 categories based on the combination of their *verb* and *target* information, as shown in Table 2. We do not consider or list the *location* information in the healing actions since it is different across different issues. In our experiments, we use the "leave-one-out" strategy, and then measure the precision/recall of our approach's effectiveness in suggesting an appropriate healing action for each "new issue". Note that in our results, the *location* info for a suggested healing action is always correct for each issue, because only an unhealthy service would produce ev1 and ev2 (see Table 1). Therefore, the retrieval accuracy is critical in the overall effectiveness of our approach.

Table 2. Categories of studied healing actions

category ID	*verb*	*target*	# of cases
ID1	reboot	SQL	26
ID2	recycle	App Pool	17
ID3	restart	IIS(authentication)	14
ID4	re-image	WFE	9
ID5	reboot	WFE(WAC)	4
ID6	patch	DB	3
ID7	restart	IIS(scanner)	2
ID8	reboot	WFE(search)	2

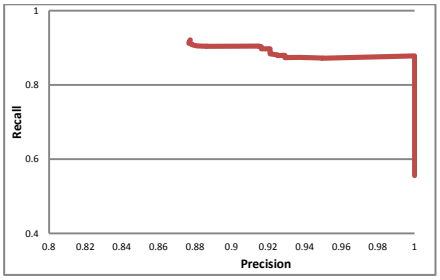

Figure 4. Overall ROC curves

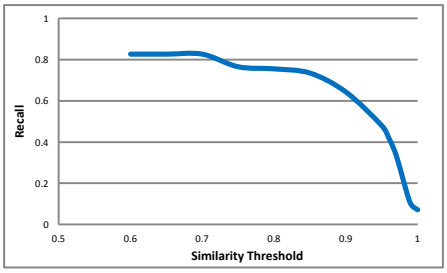

Figure 5. Similarity/recall of authentication (ID3)

4.2 Experimental Results

We illustrate our experimental results from two different perspectives. We use the similarity threshold as a parameter to draw the ROC curve of our approach. In the ROC figure (Figure 4), the X-axis denotes the precision value and the Y-axis denotes the recall value. Points located at the topper or righter side indicate higher recall values or higher precision values, being a desirable situation. Our first experimental result is the accuracy of the retrieved top 1 similar issue. The result shows that the average accuracy of our approach is 0.90. Such good accuracy is due to characteristics of the transaction logs. The transaction logs do contain "key events" that can distinguish issues from different healing-action categories, and our approach can extract these events as one signature with the highest weight. Figure 4 shows our approach's overall precision/recall. Figure 5 shows the similarity/recall curves for issues in the category "authentication" (ID3 in Table 2). Both the two results show that our approach is effective. Overall, our approach can achieve high precision and recall (Figure 4), and when the similarity gets very close to 1, we can still retrieve historical issues of correct category (Figure 5). Furthermore, the high effectiveness of our approach has been confirmed by operators for the online service system.

5. REALTED WORK

We next discuss representative related work in the areas of system diagnosis, fault localization, and mining software repositories. Cohen et al. [4] propose that retrieving a previously solved and annotated issue similar to the given new issue may help identify the root cause or fixing action for the given issue when the retrieval is accurate. In contrast, rather than aiming to fix the issue, our work aims to provide healing suggestions to reduce the MTTR by leveraging historical issues. Cellier et al. [3] apply FCA for fault localization by using concepts to find interesting clusters. In contrast to such previous technique on fault localization based on coverage information, our work is motivated by addressing challenges posed by characteristics of transaction logs. Ashok et al. [1] propose a search tool to speed up bug fixing by leveraging natural language text, dumped traces, and debugger output. Such technique would not be effective in our problem setting because the textual information in a typical historical issue repository is incomplete or imprecise.

6. ACKNOWLEDGMENT

We thank our online service system partners for their helps in verifying each case used in our experiments, and valuable comments for service and issue understanding.

7. REFERENCES

[1] B. Ashok, J. Joy, H.K. Liang, S. K. Rajamani, G. Srinivasa, and V. Vangala. DebugAdvisor: A recommender system for debugging. *ESEC/FSE'09*, pages 373-382, 2009.

[2] D. Cannon and David Wheeldon. The stationery office. *ITIL Service Operation*. ISBN 9780113310463.

[3] P. Cellier. Formal concept analysis applied to fault localization. *ICSE Companion '08*. Pages 991-994, 2008.

[4] I. Cohen, S. Zhang, M. Goldszmidt, J. Symons, T. Kelly, and A. Fox. Capturing, indexing, clustering, and retrieving system history. *SOSP' 05*, pages 105-118, 2005.

[5] S. Kandula, R. Chandra and D. Katabi. What's going on? Learning communication rules in edge networks. *SIGCOMM '08*, pages 87-98, 2008.

[6] D. Long, A. Muir, and R. Golding. A longitudinal survey of Internet host reliability. *SRDS'95*, pages 2-9, 1995.

[7] J.G. Lou, Q. Fu, S.Q. Yang, Y. Xu and J. Li. Mining invariants from console logs for system problem detection. *USENIX ATC'10*, pages 24-24, 2010.

[8] W. Xie, H.R. Sun, Y.H. Cao, and K.S. Trivedi. Modeling of online service availability perceived by web users, *GLOBECOM'01*, 2001.