ImpAPTr: A Tool For Identifying The Clues To Online Service Anomalies

Hao Wang, Guoping Rong, Yangchen Xu
wanghao.stu.nju@gmail.com,ronggp@nju.edu.cn
mf1932211@smail.nju.edu.cn
Nanjing University
Nanjing, China

Yong You
yong.you@dianping.com
Meituan-Dianping Group
Shanghai, China

ABSTRACT

As a common IT infrastructure, APM (Application Performance Management) systems have been widely adopted to monitor call requests to an on-line service. Usually, each request may contain multi-dimensional attributes (e.g., City, ISP, Platform, etc.), which may become the reason for a certain anomaly regarding *DSR* (Declining Success Rate) of service calls either solely or as a combination. Moreover, each attribute may also have multiple values (e.g., ISP could be T-Mobile, Vodafone, CMCC, etc.), rendering intricate root causes and huge challenges to identify the root causes. In this paper, we propose a prototype tool, *ImpAPTr (Impact Analysis based on Pruning Tree)*, to identify the combination of dimensional attributes as the clues to dig out the root causes of anomalies regarding *DSR* of a service call in a timely manner. *ImpAPTr* has been evaluated in *MeiTuan*, one of the biggest on-line service providers. Performance regarding the accuracy outperforms several previous tools in the same field.

KEYWORDS

Clues identification, success rate, multi-dimensional attributes

ACM Reference Format:
Hao Wang, Guoping Rong, Yangchen Xu and Yong You. 2020. ImpAPTr: A Tool For Identifying The Clues To Online Service Anomalies. In *35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20), September 21–25, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3324884.3415301

1 INTRODUCTION

The continuity and stability of on-line services are critical since even a slight decline of the success rate of service calls (abbr. *SRSC*, cf. subsection 2.2.1) may have impacted a large number of users already. In this sense, it is important to monitor *SRSC*, detect and resolve anomalies impacting *SRSC* in a timely manner.

Engineers can adopt APM tools to capture the trace of service calls, which provide valuable information to dig out the root cause behind each failed service call. However, the distributed deployment of services (e.g., microservices under DevOps context) of modern on-line service creates a phenomenon that a request call to a certain service may from various paths (traces), containing information with multi-dimensional attributes (e.g., City, ISP, Software Version, etc.). Moreover, one dimensional attribute may also contain multiple values, for example, the ISP could be T-Mobile, Vodafone, CMCC, etc. For example, a combination such as $\mathscr{S}$(5G, *ABC*, CMCC, 1.0.1, iOS) might mean the *DSR* occurs in APP version 1.0.1 when the service call is from an iOS terminal device using *CMCC 5G* network and the user is in City *ABC*. *MeiTuan* applied a self-developed APM tool (i.e., CAT[1]) to monitor traces of service calls, and each service call is stored by the "JSON" format, i.e., *{ "time": 06:00, "Network":4G, "Connect-Type": Type1, "Platform": android, "ISP": CMCC, "City": Shanghai, App-source: APP1, "App-version": 10.1.0, "code": 200 }*. As it shows, for each request, other than the values for the 7 dimensional attributes, the timestamp and status code (to indicate the success or failure of a service call) are also recorded. Obviously, there are 7 dimensional Attributes, i.e., "Network"(N), "Connect-Type"(C), "Platform"(P), "ISP"(I), "City"(Y), "App-source"(S) and "App-version"(V), respectively. And they has 5, 9, 3, 7, 310, 83 and 6700 legal dimension values, respectively. We just take the first five attributes as an example in the following sections for better understanding.

Locating and addressing root cause of anomalies are extremely important in production environment for online services. Driven by business needs, a lot of related researches have been conducted on this topic. There are two types of measures, i.e., fundamental measure (e.g., service calls) and derived measure (e.g., *SRSC*) in this topic. As listed and analyzed in study [3], several methods and tools have been proposed. HotSpot [7] and iDice [5] focuses on the fundamental measure and multi-dimensional attribute combination. Apriori [1] can deal with derived measures and the multi-dimension, but the running time is too long to adopt the real-time requirement.

Adtributor [2] focuses on both types of measures based on the forecast and real values, but it can only identify the root cause of a single dimension. One of the measures is defined as the percentage of change between the forecast and real values in the overall, and another is defined as the relative entropy between the forecast and real. As an improved version of Adtributor, R-Adtributor [6] is proposed to execute recursive calls based on the results of Adtributor. However, since the recursive depth is hard to be determined beforehand, the results may be incorrect. Further, the accurate forecast algorithm needs more history data without satisfying real-time requirement, while the simple algorithm will bring more errors.

Method Squeeze [3] is an improved version of HotSpot which is suitable to deal with derived measure, moreover, it avoids omitting

[1] https://github.com/dianping/cat

ASE '20, September 21–25, 2020, Virtual Event, Australia

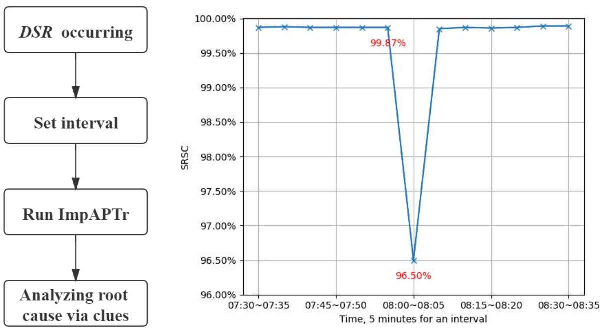
ACM ISBN 978-1-4503-6768-4/20/09…$15.00
https://doi.org/10.1145/3324884.3415301

some important attribute combinations because of the pruning strategies comparing with HotSpot. Squeeze takes the "bottom-up & top-down" to reduce the search space and locate the root-causes.

A in-depth survey study reveals that to only R-Adtributor [6] and Squeeze [3] have the potential to be adopted. However, preliminary evaluation implies that the performance of these two methods are far from satisfactory in terms of accuracy.

2 TOOL DESIGN

2.1 ImpAPTr



(a) Procedures (b) *DSR* occuring

Figure 1: The operation procedure of "ImpAPTr"

2.1.1 Procedure. In general, there are four steps (shown in Figure1(a)) to operate ImpAPTr to identify the clues of *DSR*,

(a) ***DSR* occurring.** Figure 1(b) depicts a typical example of a *DSR* of service calls occurring on March 10th, 2020. In the first interval (i.e., 07:55 ∼ 08:00), the *SRSC* is 99.87% with 824,840 service calls. However, in the second interval (i.e., 08:00 ∼ 08:05) with 1,055,240 service calls, the *SRSC* is 96.50%, indicating a *DSR* of 3.37%. All data originates from the real online services. The operation and maintenance personnel will notice the declining of *SRSC*, he needs to find out the root cause which results in the declining.

(b) **Set interval.** Due to the specified APM, the interval in our dataset is set 5 minutes, i.e., the interval 480 represents the minutes from 08:00 ∼ 08:05, therefore, the intervals mentioned above are 475 and 480.

(c) **Run ImpAPTr.** The *SRSC* of the second interval dropped by 3.37%, therefore, we should take the previous interval into consideration and calculate some measures between them. All service calls within these two intervals will be stored into two different multi-dimension arrays and run the tool "ImpAPTr".

(d) **Analyzing root cause via clues.** The results of "ImpAPTr" are several meaningful attribute-combinations (i.e., clues), while the studies mentioned in Section 1 use "root cause" to describe them. However, the real reason for an anomaly may still be covered beneath the symptoms. In this sense, we use "clues" to describe the combinations of attributes identified by ImpAPTr tool.

2.1.2 Tool demonstration. ImpAPTr is available at https://github.com/wanghaoUp/ImpAPTr. A video demonstration of ImpAPTr can be found at https://youtu.be/wJodAsezjFs As shown in Figure 1(b), it is based on the real data on March 10th, 2020 during 07:30 ∼ 08:35 . There are 13 intervals from interval 450 to 515. The *DSR* occurred on 08:00 ∼ 08:05 (i.e., the interval 480).

We run our tool by the following command,

> python ImpAPTr_test.py [day] [interval]

Therefore, for the *DSR* of March 10th, 2020 during 08:00 ∼ 08:05, we set parameter "[day]" as "10" and "[interval]" as "480", i.e.,

> python ImpAPTr_test.py 10 480

2.2 Problem Description

For better readability, we first define and clarify some key concepts and terminologies. "Dimensional attributes" can be taken as the categories of information contained in a service call which will be recorded by CAT system and "Attribute Value" represents the legal values in each dimension. Further, "Element"(e=($*$,$*$,$*$,$*$,$*$)) represents a vector of distinct attribute values.

2.2.1 Metrics. Apparently, *SRSC* and *DSR* are the key indicators in our study. To calculate *SRSC* and *DSR*, we also need some relevant metrics. We define these metrics as the following.

Combination dimensional attributes ($\mathscr{S}$) defines the set of dimensional attributes that as a combination, may contribute to an anomaly of a running service and we set up $\mathscr{S}$ as an element e, i.e.,

$$
\begin{gathered}
e = (n, c, p, i, y), \\
(n \in N \text{ or } n = *), (c \in C \text{ or } c = *), (p \in P \text{ or } p = *), \\
(i \in I \text{ or } i = *), (y \in Y \text{ or } y = *) \\
N = \{N_0, N_1, ..., N_4\}, C = \{C_0, ..., C_8\}, P = \{P_0, P_1, P_2\}, \qquad (1) \\
I = \{I_0, I_1, ..., I_7\}, Y = \{Y_0, Y_1, ..., Y_{309}\} \qquad (2)
\end{gathered}
$$

where "$*$" is a wildcard, which can free one or more constraints derived from certain dimensional attributes. Apparently, e can be taken as an instance of $\mathscr{S}$.

Success Rate of Service Calls (SRSC) measures the success rate of all service calls within a time interval under the constrain of an element e.

Declining Success Rate (DSR) measures the degree to which the *SRSC* of an element e of a certain time interval (T_i) is less than that of its previous interval (T_{i-1}), i.e., $DSR(e, T_i) = SRSC(e, T_i) - SRSC(e, T_{i-1})$. Obviously, only a positive *DSR* will attract our concerns.

2.2.2 Element Tree. To explore the reasons leading to a certain *DSR*, we have to take all dimensional attributes and their combinations as well into consideration. As discussed above, an element e is used to represent one of the combinations of dimensional attributes. However, the number of elements is thus very huge, theoretically. To portray a concept, we can construct an element tree. As shown in Figure 2, with the 5 dimensional attributes and the corresponding attribute values, we can construct a 5-layer element tree with 335 elements on the first layer, 7973 on the second layer, 69961 on the third layer, 258690 on the fourth layer and 334800 on the fifth layer, respectively. As a result, there may potentially be 671759 elements that need to be explored to identify the cause for a certain *DSR*. Apparently, if we include all of the 7 dimensional attributes in this element tree, the number of elements will explode easily.

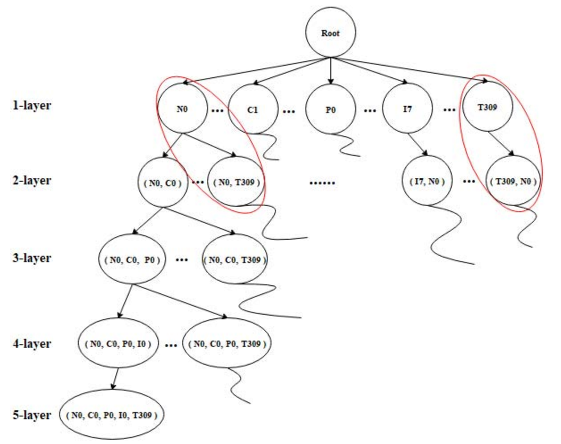


Figure 2: The element tree.

2.3 The Algorithm Inside ImpAPTr

To elaboration the algorithm, we first describe the measures, i.e., *Impact Factor*, *Contribution Power* and *Diversity Factor*, respectively.

2.3.1 Measures.

Impact Factor. Within a time interval, the degree to which a particular element (e.g. e_0) impacts the *SRSC* could be calculated as the difference between the total *SRSC* and the *SRSC* of the rest elements, which is described using *Impact Factor* (IF) as follows,

$$IF(e_0, T) = SRSC(e_2, T) - SRSC(e_1, T) \tag{3}$$

where T is the time interval.

Contribution Power. Further, we can define *Contribution Power* (Cp) as the difference of the *Impact Factor* between two adjacent time intervals (i.e., T_0 and T_1), i.e.,

$$Cp(e_0, T_1) = IF(e_0, T_1) - IF(e_0, T_0) \tag{4}$$

Diversity Factor. Diversity Factor (Df) is used to measure the change degree of the *SRSC* between two adjacent time intervals through the relative entropy calculated by the Jensen-Shannon(JS) divergence [4].

The *Diversity Factor* of the element e within the two time intervals is defined as

$$Df(p, q, e) = 0.5(p * \ln \frac{2p}{p+q} + q * \ln \frac{2q}{p+q}) \tag{5}$$

where p and q are the *SRSCs* of the two time intervals under the element e. $Df(p, q, e)$ ranges from 0 to 1. $Df(p, q, e)$ equals 0 means that no change between p and q regarding *SRSC*. Otherwise, a higher value of $Df(p, q, e)$ indicates greater change and vice versa.

2.3.2 Pruning Strategies.

Redundant elements. Since e is a combination of dimensional attributes, the order of the attribute values does not matter. Therefore, as Figure 2 depicts, two nodes (i.e., (T_{309}, N_0) and (N_0, T_{309})) in the two red cycles in 2-layer can be categorized as redundant nodes (elements). Obviously, one of the nodes and the corresponding sub-tree could be pruned from the element tree.

Positive Impacts. According to the discussion above, the *Impact Factor* represents the impact of an element on the overall *DSR*. While there might be a positive *Impact Factor* for a given element e, which means that this element e helps to decrease the *DSR*. Since we are seeking elements increasing *DSR*, we can also remove the elements with positive *Impact Factor* and their sub-trees as well from the element tree.

2.3.3 The identification algorithm.

Overall, the algorithm is a breadth-first traversal algorithm on the element tree, as shown in Figure 2. Firstly, we need to create the root node and generate its child nodes. Secondly, we applied a breadth-first traversal algorithm to fetch candidate elements which might contribute to a *DSR* (i.e., the Cp is negative according to Equation 4 and the Df is lager according to Equation 5). The generation of the element tree is along with the traversal with the pruning strategies discussed in subsection 2.3.2 to limit the size of the final element tree. Finally, we respectively record the ascending order by Cp and the descending order by Df of each candidate node. By adding Cp and Df directly, we get $rank$, which will be sorted with ascending order to generate top N possible clues.

3 EVALUATION

3.1 Dataset Preparation

To address the research questions, we evaluate the effectiveness and efficiency with the simulated dataset which is based on the base dataset directly retrieved from the production environment.

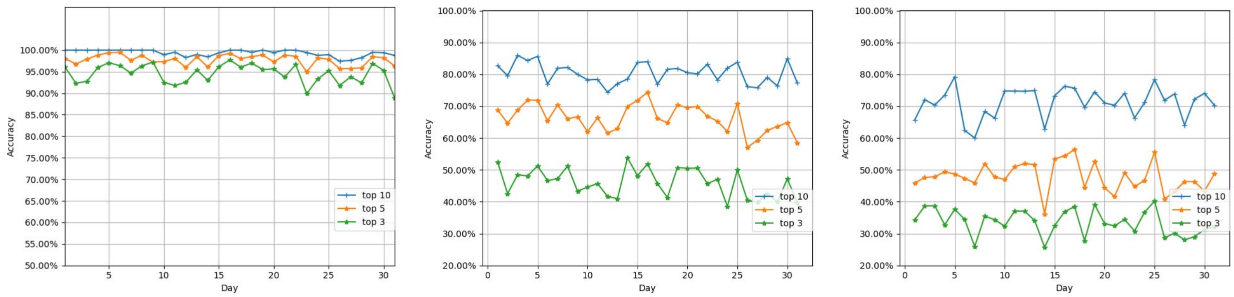
In practice, anomalies regarding *DSR* do not occur at a very high frequency. Therefore, if we intend to extensively explore the performance of *ImpAPTr*, we can not use the raw data directly. Otherwise, to include adequate anomalies, we may need to analyze the data spanning multiple months even years, which makes no sense for a service monitoring mechanism we discussed in this paper. Two points need to be emphasized.

1. Base dataset. The base dataset is extracted from the real production environment in *MT*. The monitoring system (i.e., CAT) stored the monitoring data in a database, from which we retrieved the monitoring data of the CAT from January 1 to January 31, 2020. To be specific, we retrieved the service calls to a hot service named "shop". For each day, there are around 20 to 70 million service calls to this service.

2. Anomaly planting. The object *DSR* ($SRSC_{T1}$ - $SRSC_{T0}$) is randomly selected from 0.05% to 0.1%. For each pair of adjacent time intervals, i.e., T_0, T_1, we need to plant an anomaly in time interval T_1. We first randomly choose an element e, and randomly modify the successful call to a failed call to reach *DSR*.

Table 1: The experimental data of each day.

Date	1	2	3	4	5	6	7	8	9	10	
Planting frequency	203	156	193	174	170	194	169	164	144	187	
Avg.Requests	319395	249461	282216	321834	268042	232335	241360	247066	255196	286412	
Date	11	12	13	14	15	16	17	18	19	20	
Planting frequency	207	174	196	128	155	132	152	199	177	183	
Avg.Requests	325889	288799	255659	265029	267536	270884	291163	317485	295861	276718	
Date	21	22	23	24	25	26	27	28	29	30	31
Planting frequency	176	209	178	164	188	231	210	171	194	170	162
Avg.Requests	238675	204100	139881	78043	89849	81262	80706	75915	70505	66873	63041



(a) ImpAPTr (b) R-Adtributor (c) Squeeze

Figure 3: Comparison of accuracy in top 3/5/10 results.

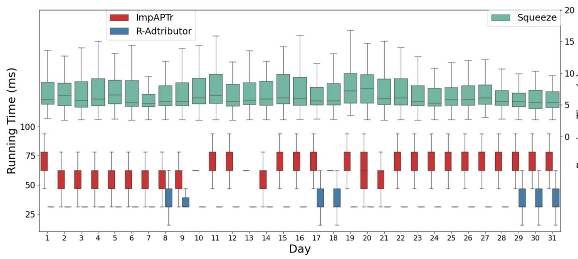


Figure 4: Locating time comparison

3.2 Result Analysis

The execution results provide adequate evidence that the approach we proposed works well to identify the possible clues leading to the root cause to anomalies. We will introduce the results from two perspectives respectively in this subsection.

Apparently, the first and foremost criteria to evaluate the performance should be the accuracy of identification of the valid clues (i.e., e in this evaluation). Therefore, we define the accuracy as

$$accuracy = \frac{SI}{SI + FI} \tag{6}$$

where SI denotes the number of successful identification of valid clues, FI denotes the number of failed identification of valid clues. However, preliminary trials implied that it was usually not possible to locate the exact one clue, we loosened the criteria of SI to top 3, 5 and 10 possible clues. Take top 3 for example, if the actual e exist in the top 3 clues after running *ImpAPTr*, then we deem it a successful identification, otherwise a failed identification. Besides, the time cost to run the identification program should also be considered to evaluate the efficiency of *ImpAPTr*.

The effectiveness of ImpAPTr. As shown in Figure 3 *ImpAPTr* performs well in terms of accuracy. For the top 3 valid clues, *ImpAPTr* presents an accuracy of 94.51% on average and ranges from 88.89% ∼ 97.73%, meaning that we can expect a ten to one correct identification. Similarly, we can expect more than 95% and nearly 100% for the accuracy if we loosen the constraints to 5 and 10 candidate clues. R-Adtributor and Squeeze did not generate satisfying results regarding the accuracy, between which R-Adtributor performs slightly better than Squeeze. But even with R-Adtributor, we can only expect the accuracy of 46.06% on average in the top 3 candidate clues.

Obviously, the performance of R-Adtributor and Squeeze are worse than our method. The main reason is that we control the DSR during 0.05% ∼ 0.1% in the dataset, while the two methods can not adapt to such a slight magnitude of DSR.

The time efficiency of ImpAPTr. Figure 4 is a box plot (with the center region from 25% to 75%) picturing the locating time of three methods. From the vertical axis on the left, we can observe that *R-Adtributor* (blue) is slightly faster than *ImpAPTr* (red). However, both methods can finish calculation within 100 millisecond, which still can be regarded as efficient, given the time interval for the data is 5 minutes. Meanwhile, from the vertical axis on the right, it could be observed that *Squeeze* is significantly slower than the other two methods.

4 CONCLUSION

For many on-line software systems with massive users, the healthiness of the software systems is critical to ensure providing services continuously. Therefore, it is important to identify and address anomalies in a timely manner so as not to impact the business. Among many indicators related to anomalies, *SRSC* and *DSR* to certain ‘hot’ services easily draw attention from the business and operation staff, yet the challenges also exist. One is the complex reasons (i.e., a combination of multiple-dimensional attributes $\mathscr{S}$) behind a *DSR*, the other is the small time slot available to find the $\mathscr{S}$, given the fact that a certain amount of time has been allocated to calculate proportional indicators such as *SRSC* and *DSR*.

Besides, the evaluation experiments of our tool are based on the datasets of MeiTuan. Theoretically, *ImpAPTr* also adapts to the different data types and datasets of other platforms, as long as the they contain different kinds of attribute information.

The tool *ImpAPTr* is developed to identify valid clues leading to the root cause behind anomalies. It can assist to identify and locate a combination of multiple dimensional attributes as the valid clues leading to the root cause of anomalies regarding a proportional indicator *DSR*. Preliminary results imply that it outperforms two previous tools in the same field. Besides, preliminary adoption of the tool in *MeiTuan* also suggests that it can help to find the root causes of anomalies which never have been identified manually.

REFERENCES

[1] F. Ahmed, J. Erman, Z. Ge, A. X. Liu, J. Wang, and H. Yan. 2017. Detecting and Localizing End-to-End Performance Degradation for Cellular Data Services Based

on TCP Loss Ratio and Round Trip Time. *IEEE/ACM Transactions on Networking* 25, 6 (2017), 3709–3722.

[2] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. 2014. Adtributor: Revenue debugging in advertising systems. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*. 43–55.

[3] Zeyan Li, Dan Pei, Chengyang Luo, Yiwei Zhao, Yongqian Sun, Kaixin Sui, Xiping Wang, Dapeng Liu, Xing Jin, and Qi Wang. 2019. Generic and Robust Localization of Multi-dimensional Root Causes. In *30th IEEE International Symposium on Software Reliability Engineering, ISSRE 2019, Berlin, Germany, October 28-31, 2019*, Katinka Wolter, Ina Schieferdecker, Barbara Gallina, Michel Cukier, Roberto Natella, Naghmeh Ivaki, and Nuno Laranjeiro (Eds.). IEEE, 47–57.

[4] J. Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (Jan 1991), 145–151. https://doi.org/10.1109/18.61115

[5] Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, and Dongmei Zhang. 2016. iDice: problem identification for emerging issues. In *Proceedings of the 38th International Conference on Software Engineering*. 214–224.

[6] Moa Persson and Linnea Rudenius. 2018. *Anomaly Detection and Fault Localization An Automated Process for Advertising Systems.* Master's thesis.

[7] Sun Yongqian, Youjian Zhao, Ya Su, Dapeng Liu, Xiaohui Nie, Yuan Meng, Shiwen Cheng, Dan Pei, Shenglin Zhang, Xianping Qu, and Xuanyou Guo. 2018. HotSpot: Anomaly Localization for Additive KPIs with Multi-Dimensional Attributes. In *IEEE Access, vol. 6*. 10909–10923.