# Many Flies in One Swat: Automated Categorization of Performance Problem Diagnosis Results

Tobias Angerstein
University of Stuttgart
Universitätsstraße 38
Stuttgart
Germany

Dušan Okanović
University of Stuttgart
Universitätsstraße 38
Stuttgart
Germany

Christoph Heger
NovaTec Consulting GmbH
Dieselstraße 18
Leinfelden-Echterdingen,
Germany

André van Hoorn
University of Stuttgart
Universitätsstraße 38
Stuttgart
Germany

Aleksandar Kovačević
University of Novi Sad
Trg Dositeja Obradovića 6
Novi Sad
Serbia

Thomas Kluge
NovaTec Consulting GmbH
Dieselstraße 18
Leinfelden-Echterdingen,
Germany

## ABSTRACT

As the importance of application performance grows in modern enterprise systems, many organizations employ application performance management (APM) tools to help them deal with potential performance problems during production. In addition to monitoring capabilities, these tools provide problem detection and alerting. In large enterprise systems these tools can report a very large number of performance problems. They have to be dealt with individually, in a time-consuming and error-prone manual process, even though many of them have a common root cause.

In this vision paper, we propose using automatic categorization for dealing with large numbers of performance problems reported by APM tools. This leads to the aggregation of reported problems, reducing the work required for resolving them. Additionally, our approach opens the possibility of extending the analysis approaches to use this information for a more efficient diagnosis of performance problems.

## 1. INTRODUCTION

Large enterprise systems are usually composed of a large number of services running on many computing nodes. Any performance degradation in these systems results in significant losses. There are many approaches for dealing with performance problems by performing diagnosis and root cause analysis, both before [10, 15] and after [5] the software enters production.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

*ICPE'17, April 22-26, 2017, L'Aquila, Italy*

© 2017 ACM. ISBN 978-1-4503-4404-3/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3030207.3030242

In large enterprise systems, application performance monitoring (APM) tools can report a large number of performance incidents, such as slow response times or increased resource consumption. However, the capabilities of these tools are limited to alerting and reporting performance problems [3], while only few approaches deal with diagnosing them [5]. In both cases, all performance problems are reported separately, again leaving the performance expert to deal with one problem at the time.

We propose to use well-known automatic categorization approaches [6] with the goal of reducing the number of performance problems that have to be analyzed. Problems that are similar to each other are categorized and dealt with together, reducing the effort required from the performance expert. We also propose to use optimization approaches, to improve the quality of the categorization result. The main goal of the preliminary work presented in this vision paper is the assessment of different clustering and optimization approaches in categorization of performance problems.

The remainder of this paper is organized as follows. Section 2 emphasizes the addressed problem and states our vision. Section 3 outlines our approach and shows preliminary findings. In Section 4 we discuss related work, while in Section 5 we draw the conclusions and outline the future work.

## 2. PROBLEM STATEMENT

In our previous work [5], we presented *diagnoseIT*—an approach for automated diagnosis of performance problems. The motivation for our work was the fact that APM practice requires enormous manual effort and expertise. As stated before, APM tools provide very little support in identifying root causes of performance problems. The manual process is often problematic for performance analysts, as most of the tasks are recurring, time-consuming, and error-prone. The goal of our work was to address this issue by formalizing APM expert knowledge. This knowledge is further used to automatically execute recurring APM tasks, such as the proper configuration of APM tools and diagnosis of performance problems, to isolate their root causes.

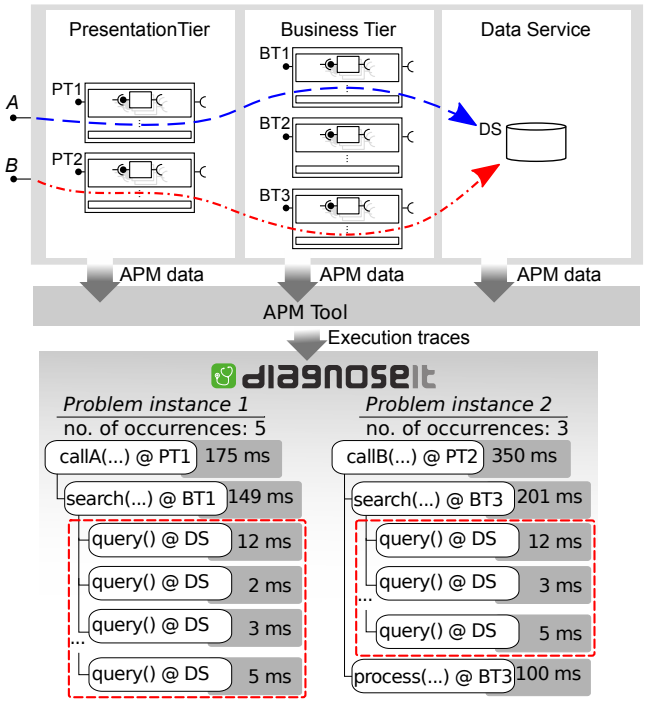


**Figure 1: Example of diagnosis results in the system that provides different services.**

This automation of the diagnosis process reduces the number of performance problems reported by APM tools, that usually have to be manually dealt with. The *diagnoseIT* approach will report where the problem occurred, provide an explanation why is this a problem, and recommend how it can be fixed. In the context of *diagnoseIT*, *problem instances* represent a concrete atomic performance problem, and are extracted from an execution trace [11]. They are described using the following attributes. *Entry point* is a point in the system from which the execution of the trace started. *Root cause* is a method in the execution trace that causes a performance problem. *Problem context* is a method which takes the largest amount of time in the execution trace, for example 80 % of a trace's execution time. *Node type* is a type of the node in the system, where the problem appears, e.g., *presentation tier*, *business tier*. *Exclusive duration* represents the time that was consumed by the affected method.

However, although there are many problem instances, they often have the same or a similar root cause. For example, consider a three-tiered system as shown in Fig. 1. This system provides two services—*Service A* and *Service B*. To process requests from clients to these two services, different components are used. While processing the requests to the *Service A*, the execution goes over the component *PT1* in the presentation tier to the component *BT1* in the business tier. In the end, *BT1* obtains the data from the service *DS*. Requests to *Service B* are processed in a similar fashion, this time using the components *PT2*, *BT3*, and *DS*.

Now consider that the component *DS* is improperly configured, e.g., the database connection pool is to small, causing the well-known N-Lane Bridge problem [13]. This results in slow response times of this component. This additional delay will cause the response times of *Service A* and *Service B* to increase. The APM tool that monitors the system will register this increase in response times as a performance problem, and it will pass it to *diagnoseIT*. The diagnosis for both performance problems is the same: "slow access to service *DS* due to an N-lane bridge". However, as they come from two different business transactions, these problems will be reported as separate problem instances. The basic implementation of *diagnoseIT* groups problem instances only if all of the attributes, except for the exclusive duration, are equal. As a consequence, in large enterprise systems, where the number of services and components is often very large, the number of reported problem instances can be extremely high. They would all have to be dealt with separately by the expert, although they all have the same root cause.

To tackle this issue, we propose a categorization of similar performance problems. This will reduce the amount of work performance experts have, as the problems with similar root causes become members of the same group. This will allow the performance expert to deal with them as one problem—"killing multiple flies with one swat."

In this vision paper, we present preliminary research on the choice of the automatic categorization approach. We perform a sensitivity analysis to evaluate the influence of problem instance attributes on the categorization result. The results of this evaluation are used for an optimization of the process. During the optimization, weights are assigned to problem instance attributes, which aims to improve the quality of the result.

## 3. CATEGORIZATION OF PROBLEM INSTANCES

An overview of our approach is presented in Fig. 2. *diagnoseIT* analyzes the data from the APM tool and generates problem instances (*PI*s). Problem instances, each represented as a vector whose features correspond to attributes of problem instances, e.g., root cause and exclusive duration, are then automatically categorized, using an unsupervised approach (Section 3.1). In this case, the resulting categories are *Category 1*, *Category 2*, and *Category n*. Along with the process of categorization, optimization is performed (Section 3.2). The initial categorization of problem instances is performed using default weights. The role of weights is to define the influence of each problem instance attribute on the categorization result. The goal of the optimization is to generate a better set of parameters for categorization by as-

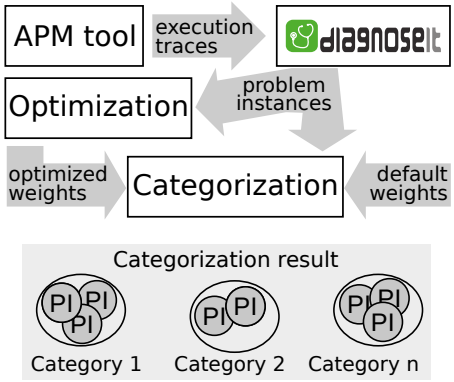


**Figure 2: An overview of the approach.**

signing different weights to attributes of problem instances, as these weights can be specific for each case.

Section 3.1 provides an overview of the considered automatic categorization approaches. Section 3.2 provides an overview of the optimization approaches. Preliminary findings are presented in Section 3.3.

## 3.1 Categorization of Problem Instances

Cluster analysis is an unsupervised approach to organizing data elements into clusters based on their proximity [6]. Contrary to supervised approaches, unsupervised approaches do not require pre-classified training data. In our case, since we want to perform the analysis automatically and do not have prior information about the data, we opt to use an unsupervised approach.

According to Jain et al. [6], there are several steps in the clustering activity. The first step is pattern representation. A problem instance can be represented as a vector $\vec{P} = (p_1, p_2, ..., p_n)$, where $p_1$, $p_2$,..., $p_n$ are the values of the problem instance's attributes (as described in Section 2).

In the second step, proximity of patterns is defined. Our approach uses Euclidean distance because of its simplicity. However, Euclidean distance can be very sensitive to distortion, e.g., when data does not consist of compact and isolated clusters. A solution for this is to assign weights to features, so that the vector $\vec{P}$ becomes $\vec{P'} = (w_1 \cdot p_1, w_2 \cdot p_2, ..., w_n \cdot p_n)$.

The third step is the choice of the most appropriate approach for grouping of patterns, and in our case, it is dictated by the following assumptions.

*Uniqueness* Each problem instance represents one performance problem, so each problem instance should belong to one category only.

*Independence* The algorithm should be independent of the number of categories, as the potential number of categories is not known at the beginning of the process.

*Performance* The algorithm should be able to handle large numbers of instances in a reasonable amount of time.

We found that, among different clustering approaches [1, 6, 16], *hierarchical clustering* and *k-means algorithm* satisfy our requirements. Other approaches are not suitable because they provide multiple solutions, e.g., evolutionary clustering, or require some sort of training set, e.g., artificial neural networks.

**Hierarchical clustering** provides the result as a hierarchical structure. It is independent of any initial choice of clusters, and provides one solution. However, the drawback of this approach is that it can be very slow because of its high complexity.

**k-means algorithm** is faster than hierarchical clustering, as it has fairly low complexity. Although the number of clusters has to be known in advance, there are ways to perform the estimation [14].

The obvious advantage of using the k-means approach is its speed. On the other hand, the hierarchical approach provides results that can be useful in the diagnosis of performance problems, as problem instances can be categorized by symptoms [15].

Assessing the quality of clustering result can be done in two general ways [12]. Using the *internal quality criteria*, the clustering result is evaluated based on the data about clusters themselves, such as the distances between instances in the cluster (*Sum of Squared Errors*) and the distances between clusters (*Standard Deviation of Cluster Center Distances*). *External quality criteria* use external data, such as separately created reference clusters.

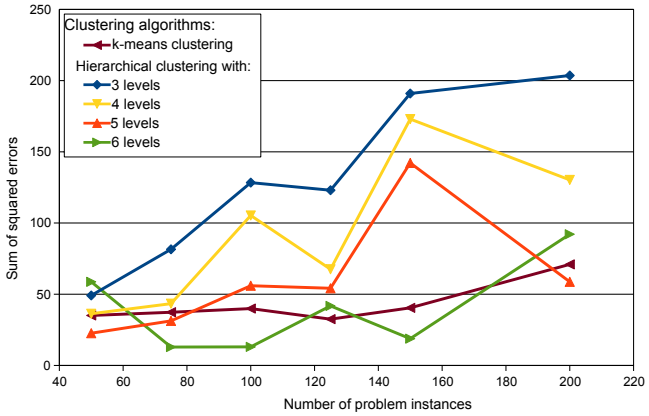


**Figure 3: Sum of squared errors for k-means and hierarchical clustering algorithms, for different number of problem instances.**

## 3.2 Optimization of Categorization

As stated before, not all problem instance attributes contribute equally to the clustering result. To assess the influence of each attribute, a calibration of parameters is needed to assign them weights.

We evaluated three approaches for optimization. The *evolutionary approach* uses standard evolutionary operators to create variations in weights. After clustering is performed with each set of weights, results are compared with a fitness function. The *hill-climbing* constantly modifies weights as long as the result is improving. The *human-in-the-loop* approach requires manual effort, instead of the fitness function, to pick the optimal result.

## 3.3 Preliminary Evaluation of Approaches

A preliminary evaluation of the selected approaches was performed using the data obtained by monitoring the DVD Store application and the inspectIT APM tool,[1] which was then analyzed with *diagnoseIT*. The problem instance categorization and optimization was performed using the Weka framework [4]. The experimental setup and a runnable example can be found in the supplementary material.[2]

An evaluation of the clustering results using internal quality criteria shows that the results of the k-means algorithm are better, although, in some rare cases, hierarchical clustering has some advantage (Figure 3). Similar results are obtained when analyzing the standard deviation of cluster center distances. Additionally, results show that, as expected, k-means performs faster. Although these results are as expected, hierarchical clustering proves valuable for the diagnosis of performance problems, because of the hierarchical structure of the result.

A comparison of optimization approaches has shown that the results of automatic optimization approaches almost do not bring any improvement of clustering results. However, improvement was present with the human-in-the-loop ap-

[1] http://www.inspectit.rocks/

[2] https://doi.org/10.5281/zenodo.321383

proach, which leads us to the conclusion that some knowledge about the observed system should be included. Although this seems contrary to our initial idea, inclusion of humans in this case is justified, as the performance analyst would deal with a much smaller number of problem instances.

# 4. RELATED WORK

To the best of our knowledge, modern APM tools [3] provide no aggregation of performance problems. Some tools report the total number of incidents, e.g., how many times response times exceeded a threshold, or aggregate some of the metrics, e.g., average response time of some method, but the correlation between incidents has to be established manually.

There are some approaches to dealing with the large number of results in load testing. Malik et al. [9] use Principal Component Analysis to reduce the performance counter data, e.g., CPU utilization, disk I/O, queues and network traffic, and identify performance gains and losses. The same group of authors [8] proposes the use of performance signatures, which aggregate the data from performance counters into smaller, more manageable sets. Further, to deal with these sets, they propose both supervised and unsupervised approaches (including k-means) to detect possible performance deviations. The reduction of data is a topic of Foo [2], where performance metrics are converted into discrete levels. However, this results in loss of information about small deviations between these levels. The approach proposed by Jiang [7] uses execution logs and performs mining of the load test data. The drawback of the approach is that it relies on the log formats, which are platform-specific.

# 5. CONCLUSIONS

In this paper we presented an approach to categorization of problem instances for more efficient problem diagnosis. The goal is to group similar problem instances, and reduce the effort of manually inspecting every single problem instance that is required by performance experts. Our preliminary results are promising, as the use of k-means provides a manageable number of clusters. The use of hierarchical clustering, although slower, provides results that are particularly useful for performance problem diagnosis.

Future work will deal with the extending of the problem instances, so that they include more data from the execution trace, e.g., which classes and components were involved, on which nodes the execution took place, and timing information about all segments of the execution. We think that this would allow us to detect more complex performance antipatterns that cannot be detected from a single trace. Also, we plan to investigate the influence of problem instance attributes. Furthermore, we plan to investigate other categorization approaches.

# 6. ACKNOWLEDGMENTS

This work is being supported by the German Federal Ministry of Education and Research (grant no. 01IS15004, *diagnoseIT*), the Research Group of the Standard Performance Evaluation Corporation (SPEC), and by the Serbian Ministry of Education (project "Infrastructure for Technology Enhanced Learning in Serbia (III47003)").

# 7. REFERENCES

[1] V. L. Brailovsky. A probabilistic approach to clustering. *Pattern Recogn. Lett.*, 12(4):193–198, 1991.

[2] K. C. Foo. Automated discovery of performance regressions in enterprise applications, 2011.

[3] C. Haight and F. D. Silva. Gartner's magic quadrant for application performance monitoring suites, 2016.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009.

[5] C. Heger, A. van Hoorn, D. Okanović, S. Siegl, and A. Wert. Expert-guided automatic diagnosis of performance problems in enterprise applications. In *Proc. 12th Europ. Dependable Computing Conf. (EDCC '16)*, 2016.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[7] Z. M. Jiang. Automated analysis of load testing results. In *Proc. 19th Int. Symposium on Soft. Testing and Analysis (ISSTA '10)*, pages 143–146, 2010.

[8] H. Malik, H. Hemmati, and A. E. Hassan. Automatic detection of performance deviations in the load testing of large scale systems. In *Proc. 35th Int. Conf. on Soft. Eng. (ICSE '13)*, pages 1012–1021, 2013.

[9] H. Malik, Z. M. Jiang, B. Adams, A. E. Hassan, P. Flora, and G. Hamann. Automatic comparison of load tests to support the performance analysis of large enterprise systems. In *Proc. 14th European Conf. on Soft. Maintenance and Reengineering (CSMR '10)*, pages 222–231, 2010.

[10] A. Nistor, P.-C. Chang, C. Radoi, and S. Lu. Caramel: Detecting and fixing performance problems that have non-intrusive fixes. In *37th Int. Conf. on Soft. Engineering (ICSE '15)*, pages 902–912, 2015.

[11] D. Okanović, A. van Hoorn, C. Heger, A. Wert, and S. Siegl. Towards performance tooling interoperability: An open format for representing execution traces. In *Proc. of the 13th European Workshop on Perf. Engineering (EPEW '16)*, pages 94–108, 2016.

[12] L. Rokach and O. Maimon. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

[13] C. U. Smith and L. G. Williams. Software performance antipatterns. In *Proc. 2nd Int. Workshop on Soft. and Performance (WOSP '00)*, pages 127–136, 2000.

[14] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, 63:411–423, 2000.

[15] A. Wert, J. Happe, and L. Happe. Supporting swift reaction: Automatically uncovering performance problems by systematic experiments. In *Proc. 2013 Int. Conf.on Soft. Engineering (ICSE '13)*, pages 552–561, 2013.

[16] C. T. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comput.*, 20(1):68–86, 1971.