Multi-level Event And Anomaly Correlation Based on Enterprise Architecture Information

Jörg Landthaler, Martin Kleehaus, and Florian Matthes

TU München,
Boltzmannstr. 3, 85748 München, Germany
{joerg.landthaler,martin.kleehaus,matthes}@tum.de
http://wwwmatthes.in.tum.de

Abstract. Growing IT landscapes in and among enterprises face the challenge of increasing complexity, which complicates root cause analysis and calls for automated support. This paper presents an approach to correlate events, e.g. anomalies in multi-level monitoring stream data, for instance conversion rates or network load monitoring. Events, e.g. operational activities like application deployments and marketing activities can be taken into account, too. We exploit an Enterprise Architecture documented as a graph to focus on those correlations, where relationships are already known. Therefore, different data source types are identified. We present a minimal prototypical implementation called MLAC that shows first results of the feasibility of the approach, in particular to correlate events and level shift anomalies in an artificial web-shop setup. It includes a dynamic visualization of the correlations in the EA graph.

Key words: event correlation, anomaly detection, outlier detection, stream data types, anomaly types, enterprise architecture, multi-level monitoring, runtime monitoring, time series data

1 Introduction

The management of large IT landscapes is a big challenge for enterprises. Digitalization is currently recognized as a major trend. Therefore, IT landscapes of enterprises can be assumed to become even larger and more complex, also due to the interconnection of IT landscapes from different enterprises. One key challenge is to identify root causes for problems occurring within these IT landscapes. This is difficult, because often a bunch of monitoring tools is used, which complicates the correlation of events. Moreover, the effects of problems sometimes emerge in multiple key performance indicators (KPIs), e.g. in monitoring data recorded from business processes, applications, hardware and networks. Thus, a monitoring solution covering all sources is desirable. In addition to that, our hypothesis is that the dependency information, such as captured by Enterprise Architecture Management (EAM) tools, could facilitate the identification of root causes.

A layered perspective on monitoring solutions is not new, cf. [1]. Similar to that a standard Enterprise Architecture Management (EAM) model distinguishes between business, application and infrastructure layers, where each performance indicator can be assigned to. Moreover, events constitute an efficient way of dealing with relevant information from time series data and has been used by several researchers, cf. Section 2. Our idea is to perform anomaly detection on each of the performance indicators and to correlate changes of different performance indicators and also events induced by operational activities on the basis of the information contained in EA models. An Enterprise Architecture (EA) model can be used to document dependencies of performance indicators of different EA layers *a priori* and is often maintained by EA specialists for their own needs. This can be exploited to focus first on correlations, where relationships are already documented. We assume that the correlations detected by this approach speed up the process of root cause analysis, because there is already a documented dependency. Figure 1 illustrates the idea of our approach. Furthermore, approaches exist to discover these dependencies automatically, known as EA discovery, that could be used to automate the process of the graph generation at least to some degree.

A simple use case to illustrate the applicability of our approach could be for example the conversion rate as a KPI in the business layer of an EA that depends on multiple servers. On the servers the corresponding web-shop application or parts of it (e.g. load-balancer, web-servers, databases) are deployed and running. Here, several standard performance indicators of servers could be used to detect erroneous behavior of the server, e.g. ping results, cpu load, network bandwidth and alike.

This work is part of the integrated multi-level monitoring part-project [2], [3] of the TUM Living Lab Connected Mobility (LLCM) project. The interdisciplinary TUM LLCM project [4] envisions an open platform to support and provide new mobility concepts. For this open platform central integrated monitoring services shall be developed including a capability to logically connect monitoring information from different layers of the EA model. However, our general-purpose approach can be applied to a multitude of application scenarios.

The key contributions of this work are a new approach to detect correlations of different types of anomalies and events occurring in an IT landscape based on information gathered from an enterprise architecture and a minimal viable prototype called MLAC to demonstrate the feasibility of the approach. Therefore, we also identify different data source types and discuss potential anomaly detection methods and further suitable events.

The remainder of this paper is organized as follows: Section 2 shortly reviews relevant prior and related work. Our general idea of multi-level anomaly correlation (MLAC) is described in Section 3. In Section 4 we identify different types of data streams and their anomalies and changes. The prototypical implementation is described in Section 5. In Section 6 we discuss limitations of both: the

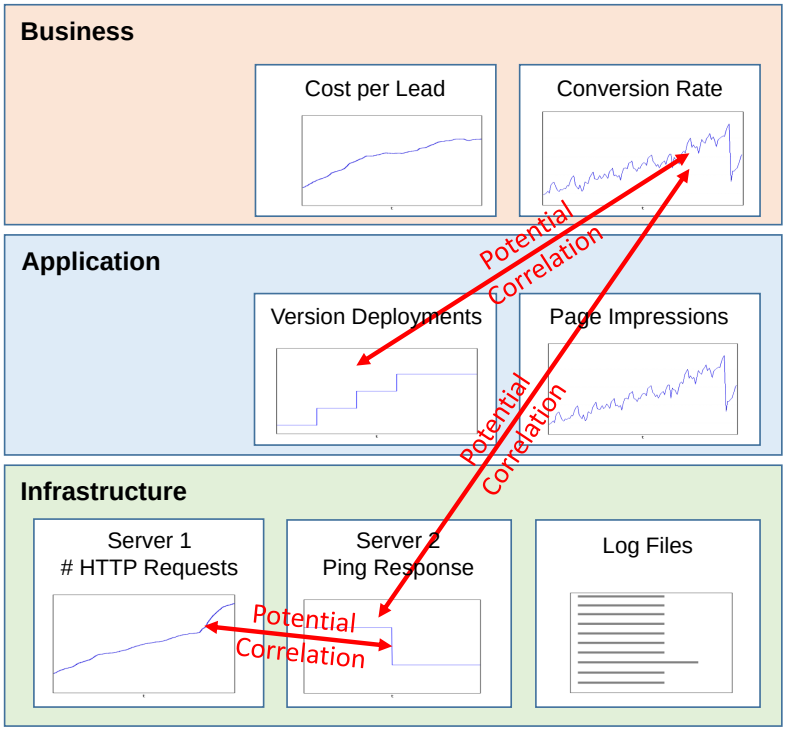

Fig. 1. Illustration of our approach for the correlation of anomalies in KPI streams (monitoring data) and events among different layers of an enterprise architecture model. Correlations can be detected among anomalies and anomalies, anomalies and events and on events and events within one or across multiple layers of the EA.

idea and our prototype. We also present ideas for future work. Finally, Section 7 briefly concludes this paper.

2 Related Work

Time series data typically occurs in the domain of monitoring. A plethora of work has been done on monitoring KPIs for each of the EA layers, including extensive reviews, e.g. on the topics of business process monitoring [5], network monitoring, e.g. [6] or application monitoring, e.g. [7]. Several monitoring approaches monitor data from two layers, e.g. grid monitoring [8] or eventually even three layers, e.g. cloud monitoring, see e.g. [9].

A handful of approaches attempt to combine event data extracted from monitoring data from several EA layers. For instance Zeginis et al. [10] collect monitoring data from all EA layers, but they do not correlate the data. Vierhauser et al. [11] propose the ReMinds framwork that uses complex event processing to

correlate events based on monitoring data, but only uses monitoring data from the infrastructure and application layers. Baresi et al. [12] and Mos et al. [13] correlate events from all EA layers. However, none of these approaches exploits the structural information among the entities in the layers.

The domain of detecting outliers and anomalies has a long history. Even the very first definition of an outlier dates back to 1980, and is given by Douglas M. Hawkins [14] "An outlier is an observation that differs so much from other observations as to arouse suspicion that it was generated by a different mechanism." In the following decades, especially in the domain of time series forecasting many methods have been developed, e.g. robust regression [15], ARMA models [16] [17] [18] [19], and ARIMA models [20] [21], mostly for forecasting purposes. They have been used for anomaly detection later on.

Albeit anomaly and outlier detection can be applied on several data types, we only focus on work capable of using time series data. Analogue to the time series forecasting domain many methods are designed for specific application domains. Aggarwal [22], Chandola et al. [23] and Hodge et al. [24] provide an extensive overview of outlier detection techniques spanning multiple research areas and application domains. Gupta et al. [25] provide an extensive overview of anomaly detection for temporal data like time series data, data stream, distributed data, spatio-temporal data or network data. Ranshous et al. [26] compiled a considerable survey about anomaly types that occur in dynamic networks and introduced five methods how to detect these anomalies. A further survey conducted by Akoglu et al. [27] comprises not only anomalies types and detection methods in dynamic networks but also provides an extensive overview about static, dynamic, attributed and plain graphs. Furthermore, the authors highlight the effectiveness, scalability, generality, and robustness aspects of the introduced methods and address major techniques that facilitate the root cause analysis of the detected anomalies. We restrict ourselves to a suitable amount of types for the illustration of our approach.

Finding the root causes of anomalies has been extensively studied in various domains. In computer networks, effort has been made on handling both real-time events and link information by means of an extension of the Principal Component Analysis (PCA) algorithm [28], or by the so-called hierarchical domain-oriented reasoning mechanism [29]. Yan et al. [30] present an approach to determine the anomaly localization by passively monitoring the end-to-end performance associated with end-users from inside an Internet Service Provider (ISP) network. The authors of [31] describe a fault localization methodology in an end-to-end service system by using belief networks. The graph models the dependency among network components layered in a multi-level system. Each layer is associated with multiple possible failure modes. After detecting anomalies in the system, belief propagation algorithms are running on the graph and the posterior beliefs are examined to pick out the most likely causes for the anomalies.

Although these aforementioned techniques are good approaches to determine the root cause of anomalies by incorporating multiple data flow layers, they

provide only a very inaccurate investigation of the troubleshooting and does not consider the business processes on top of the described network. The developers of the large-scale monitoring systems VScope [32] and Monalytics [33] attempt to efficiently find the root causes of anomalies by considering the relationships between components and sensors. In contrast, our approach attempts to find the root cause of anomalies by comparing simultaneously occurring outliers in time series data and events collected from several enterprise architecture layers.

Many proposals leverage machine learning and data mining techniques for finding root causes in systems. One approach is to learn from historical data and find anomalies in the current data. [34] presents a decision tree learning approach to diagnose failures in large internet sites. The authors of [35] uncover root causes of failures in sensor network applications by performing discriminative frequent pattern mining based on frequent patterns generated by the so-called Apriori algorithm. Both methods employ supervised algorithms, whereas Kim et al. [36] introduce an unsupervised model for finding the cause of anomalies in a service oriented web architecture. The model combines historical and latest service metric data to rank sensors that are potentially contributing to a given anomaly. All listed methods do not attempt to find the root cause of anomalies by detecting correlations between time series data created by means of relevant KPIs based on the particular enterprise layer.

From the wide variety of outlier detection techniques available, we choose the BIRCH algorithm proposed by Zhang et al. [37]. BIRCH is an unsupervised stream data clustering method based on balanced trees. It has notable advantages in comparison to other algorithms: It performs effectively over large data sets, which is often the case in high frequency time series data and it produces good clustering from only one scan of the entire data set.

3 Multi-Level Anomaly and Event Correlation (MLAC)

Our approach attempts to find potential correlations of events in an IT landscape. The main assumption for our approach is that anomalies or events that are causally linked occur together within a certain time frame. Events can be anomalies detected in monitoring time series data or events encoding operational activities, e.g. version deployments or marketing activities. Therefore, each data source is considered as a time series (monitoring data and operational events), see also Section 4.

After the first processing step only events (anomalies detected in time series data or operational events) are considered. For each event it is assessed, whether other events occurred within a reasonable time frame. The total number of possible correlations is reduced by taking into account the information extracted from an EA, i.e. possible correlations are considered only if a path between the different data source layers exists in the graph representing the EA.

Within each time step of our algorithm, a limited amount of data is collected for each data source. Subsequently, the time series pre-processing / change detection is performed and the events are assessed for possible correlations according

to the EA graph. Moreover, a time window describing the maximal distance in time for the possible correlation of events can be set. This process is repeated for each time step.

It is possible to detect correlations among events only, anomalies detected in time series data and events and also among anomalies stemming from different time series data only, with the latter being of course the most interesting combination. This approach requires the user to maintain a correct and possibly large EA graph on the one hand. On the other hand, it allows a large flexibility with respect to the different data source types.

4 Data Source Types and Anomaly Detection Methods

Our approach takes into account anomalies detected on time series data as well as special events. There are two views on this. Events can be imagined as time series data with only binary or ordinal values, as depicted in Figure 3. Of course, they can be saved much more efficiently by only saving the time stamp and the value when changes occur. Similarly, anomalies can be imagined as events in time series data. This relationship of the input data is depicted in Figure 2. Consequently, the input to our algorithm can always be assumed to be a time series and the output of the anomaly detection or event detection methods can always be assumed to be events (except for log data).

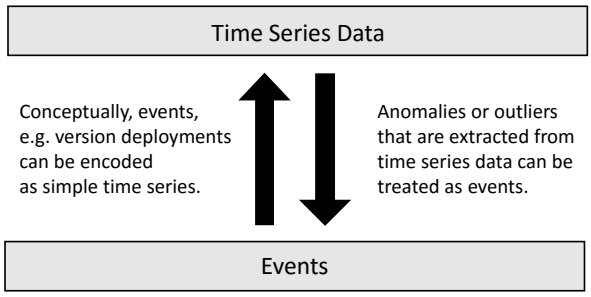

Fig. 2. Illustration of the relation between different possible inputs to our approach.

Nevertheless, there is a multitude of events in time series data that can be relevant for the detection of possible correlations in the IT landscape. In the literature different types of anomalies in time series data have been identified, e.g. by [38] [21] [39]:

– **additive outlier (AO)** affects only a single observation and jumps back to the normal behavior after the anomaly occurred. Additive outliers can adopt a seasonal characteristic which appears as a surprisingly large or small value occurring repeatedly at regular intervals.

- **innovative outlier (IO)** presents an unusual innovation in the time series and affects all later observation. In detail the observation can increase, decrease or follow a constant shift. A mixed innovative outlier can be observed when only one characteristic of the time series, e.g. trend or seasonal component, is changed by the innovation.
- **level shifts (LS)** characterize outliers which experience a move in their original level. Like an innovative outlier LS anomalies affect many observations and have a permanent effect.
- **temporary changes (TC)** describe an anomaly which produces an initial effect that dies out gradually with time.
- **reallocation outlier (RO)** introduced by Wu [39] affects a series of coherent additive outliers. In addition, reallocation outliers exhibit the particular characteristic that all the affects add up to null.

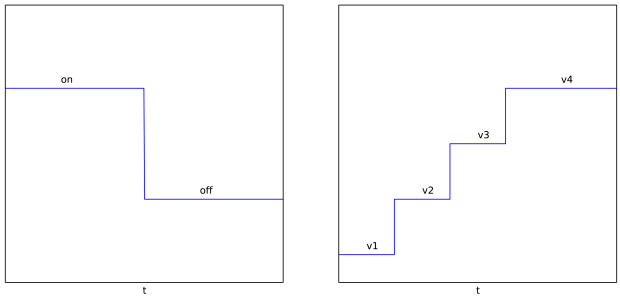

Fig. 3. Binary (left) and ordinal (right) event types encoding e.g. server ping responses, version deployments or marketing activities (e.g. TV spots)

For our approach, the user needs to manually decide what type of anomalies or events are relevant for each data source. Therefore, we classify a selection of the possible events that can occur in the IT landscape:

- Events that can be encoded in **binary or ordinal valued time series**, e.g. server ping responses, version deployments or marketing activities: Figure 3.
- Simple events in real-valued time series data, e.g. exceeding an **upper bound threshold** or falling below a certain **lower bound threshold**, see Figure 6 for an illustration.
- Anomalies extracted from linear or cyclical time series models using **anomaly or outlier detection methods**.
- **Complex time series data** can be reduced to trend, cyclical, seasonal and noise components using time series decomposition. The trend and cyclical components can then be fed into anomaly and outlier detection methods for linear or cyclical time series data.
- Events extracted from semi-structured or structured **text documents**, e.g. log files.

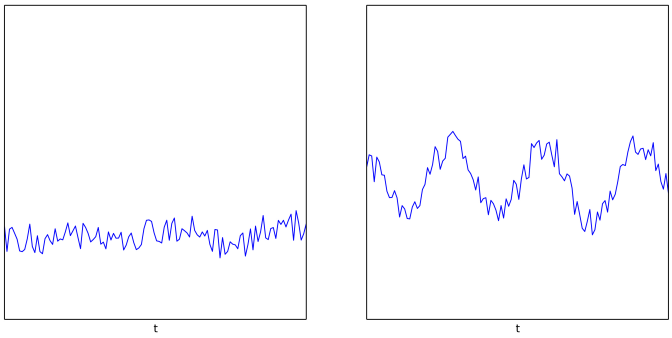

Fig. 4. Trend (left) and seasonal or cyclical (right) time series data modeling for example conversion rates or network load monitoring data.

Some events in the IT landscape can be encoded simply as binary on/off values, e.g. server ping responses. Marketing activities and relatively sparse events in time can easily be encoded as ordinal time series values. The automated detection of outliers in real-valued time series data is much more challenging. There are several types of anomalies. As already discussed there are single outliers as well as qualitative changes. The latter might be more relevant for concrete use cases. Furthermore, there exists a plethora of different algorithms for anomaly and outlier detection on time series and stream data. For each data source a specific use case requires one or several different types of anomalies to be detected.

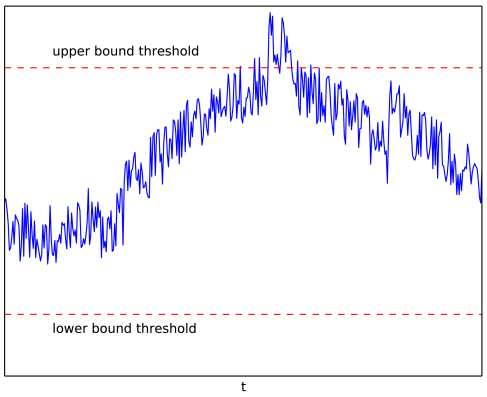

Fig. 5. Upper and lower bound thresholds on time series data that encodes e.g. preventive alerts on network bandwidth load monitoring data.

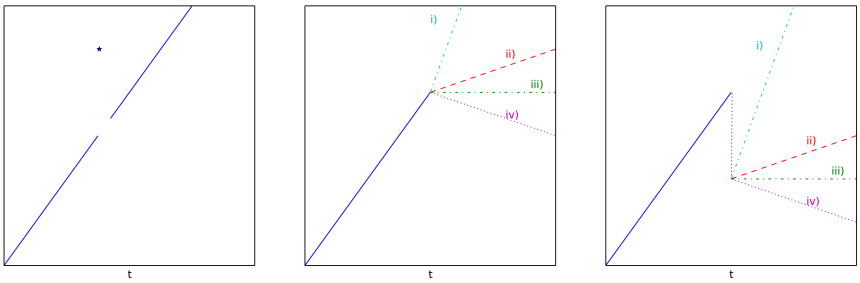

Fig. 6. Illustration of different anomaly types extracted from time series data known in the literature, e.g. additive (left), innovative (middle) and level shift (right). Most relevant for a conversion rate time series are innovative anomalies. However an increasing growth depicted in i) can be desirable, whereas smaller decreases ii) or a stopping of growth could be considered normal, and iv) switching to a decrease in growth could be considered as a relevant anomaly. Additionally, these changes in the trend can be preceded by a significant drop or rise in the values of the time series.

5 The MLAC Prototype

The Multi-level events and anomaly correlation (MLAC) prototype is currently designed as a workbench to evaluate the feasibility of our approach. The overall architecture of MLAC and the data flow among its components is depicted in Figure 7. In general, it follows a model-view-controller pattern approach, sepa-

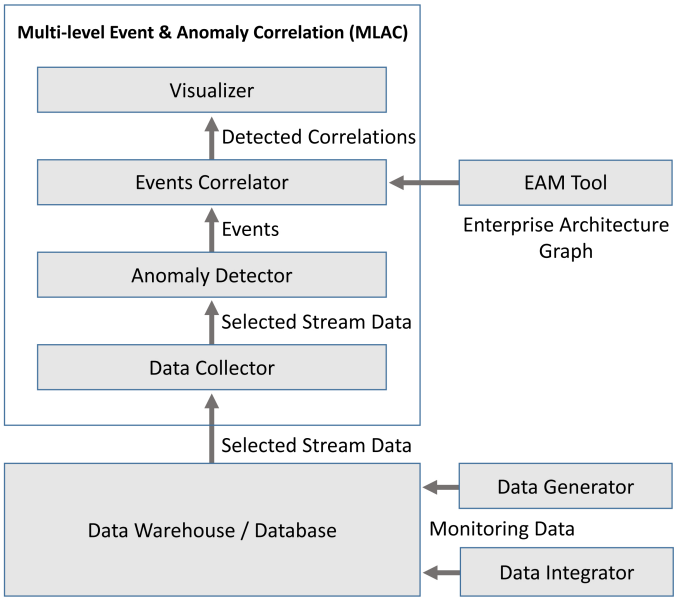

Fig. 7. Illustration of the architecture of the MLAC prototype and its data flows.

rating data management, processing of data and visualization of results. Within the processing part it can be divided into two sub parts: pre-processing of the time series data and the correlation of events. One result of this division into components is that each component can rather easily be improved or replaced. The individual components of the MLAC prototype are described in the following:

- **Database / Data Warehouse, Data Integrator, Data Generator:** Currently, we generate artificial streaming data in a Data Generator component and store it in a HSQLDB database. For real world applications a time series database or a data warehouse to keep data is essential, e.g. Pentaho that collects data using a Data Integrator component from various sources, e.g. from marketing analysis tools like Google Universal or Piwik.
- **Data Collector:** This component of MLAC is an interface to access relevant time series data from a database or adData warehouse. It queries the data store and selects only time series data within the configured time frame.
- **Anomaly Detector:** The quality of the proposed correlations heavily depends upon the quality of the detected anomalies or changes. Therefore, the Anomaly Detector component provides an interface to invoke a suitable anomaly detection algorithm on the time series data.
- **Anomaly Correlator:** The core component of MLAC is the Anomaly Correlator. It runs in an endless loop, where within each loop the algorithm matches all detected anomalies with respect to the edges in the EA graph and time constraints.
- **EAM Tool:** The EA architecture in an enterprise context is often documented in an EAM tool, e.g. ArchiMate or alike. Currently, the graph is directly modeled in MLAC, but a component that imports the graph information from EAM tools is planned.
- **Visualizer:** The visualizer component depicts the EA as a graph and highlights edges.

In order to perform a first evaluation of the feasibility of our approach, we choose a very simple and artificial setup. The enterprise architecture setup of a simple web-shop example is depicted in Figure 8 b): Ten servers on the infrastructure level contribute to two different business level KPIs (conversion rates). Five servers contribute linearly to each of the conversion rates. The server availability is simulated and measured as simple binary time series (cf. Figure 3) representing the server ping responses.

If one server is not running, the conversion rate will drop for one fifth of its normal value. Note that multiple servers can shut down in parallel. The conversion rate hence contains level shifts and we perform an outlier detection using the BIRCH algorithm on the conversion rates. We use the common heuristic to consider the smallest cluster to contain most likely outliers. The idea is that the detection of changes in the server ping responses is sharp. This is done to evaluate if the BIRCH algorithm is a suitable algorithm to detect level shift changes in the conversion rates. Figure 8 illustrates the textual and visual output of our prototype.

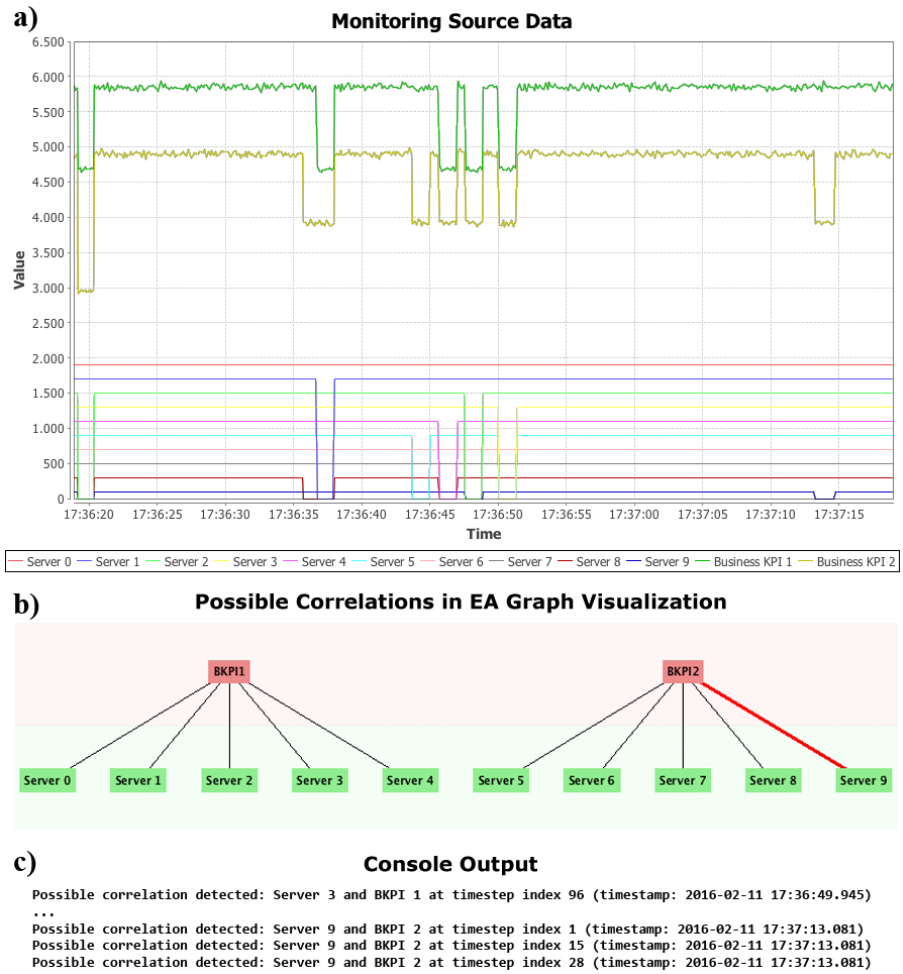

Fig. 8. Screenshots of the prototype's different outputs: a) Illustration of source data streams (multiple time series depicted in one plot for a clear graphical presentation), b) Graphical illustration of the detected correlations in the EA model depicted as a graph: Correlations indicated by bold red edges, c) Textual output of detected correlations. In this example two business KPIs (conversion rates) are plotted in the upper part of a) and the lower of the two drops around 17:37:14 for about one second. At the same time server 9, the bottom line in plot a) fails to answer. For this artificial example our approach is capable of detecting the correlation, which is displayed textually in the three bottom lines of c) and also indicated by the bold red line in b). Note that [40] is used for b).

Using this simple artificial setup, we can see that the BIRCH algorithm needs to be fine-tuned using the distance threshold parameter that controls the diameter of clusters and indirectly also controls the total possible number of clusters for a particular data set. It is necessary to find a value in a range so that the BIRCH algorithm results neither in a single large cluster nor in as many cluster centers as there are values. Despite these manual needs for fine-tuning, the BIRCH algorithm is capable of detecting the points in time, where level shifts occur. Using the heuristic that the smallest cluster center contains the outliers, it is not possible to detect zero outliers (if there are none in human terms). We use a value that detects more outliers than necessary. This is due to the simple setup. Moreover, using the EA graph we can filter out unnecessary outliers, because there are no other events in the system at the same time.

So far a simple evaluation with a toy setup shows that the approach is feasible in general and that a fine-tuning of the distance threshold parameter influences the quality of the detected anomalies. It is possible to tune it in a way that relatively simple level shifts in a time series with small noise can be detected and that a small over-fitting can be accepted because of the sharp nature of the events of the toy example.

6 Limitations and Future Work

A current limitation of the prototype and subject to future work is that changes are not correlated among disjoint time steps, i.e. a change that occurs a short time period before or after a change on a different stream are not considered to be correlated. This can be resolved by a window of a fixed number of time steps around the currently considered time step. However, it would increase the total number of necessary comparisons by the number of considered time steps. Moreover, a current limitation is that the data streams need a joint time axis. A support for variable time axis granularity is planned.

Currently the prototype is build purely in Java. It should be extended to be accessible and configurable via a web-based frontend in the future. Additionally, the data should be stored in either a time series database or a data warehouse like Pentaho [41]. A time series database has the advantage of a highly optimized software for storing and querying time series data. In contrast to that Pentaho has the advantage of easy integration of existing data sources. Moreover, the integration of a complex event processing framework, e.g. Drools Fusion [42] could simplify and empower the configuration of the prototype.

A major limitation of our approach in general is that it depends heavily on a properly maintained enterprise architecture documentation (the EA graph) and the quality of the detected anomalies and changes. With respect to the quality of detected anomalies several improvements can be thought of: Anomalies depend on the size of the sliding window, i.e. the number of data points considered. Hence, the anomaly detection methods could be performed on different window sizes and a subsequent majority voting could improve the anomaly detection. Also, machine learning methods could be applied on the resulting anomalies to

improve the quality of the anomaly detection. Last but not least, several different methods like BIRCH and AnyOut could be applied simultaneously on the same data in order to detect anomalies of higher quality.

As future work we intend to experiment with different anomaly detection algorithms, especially after the application of time series decomposition algorithms on real world data.

7 Conclusion

The MLAC prototype correlates changes in monitoring data streams from different elements in enterprise architecture. A major benefit is the reduction of necessary comparisons by comparing only anomalies or changes along the edges of a graph encoding an enterprise architecture. Anomalies can be detected with any anomaly or outlier detection algorithm. Therefore, we identified different types of anomalies from a use case point of view. Moreover, we can extend the approach to encode changes or activities like software deployments or marketing activities as monitoring data streams. We achieve promising results with our first minimal viable prototype. It encodes server availability of multiple servers that linearly influence fictive business KPIs. We use a BIRCH implementation to detect anomalies on the business KPI monitoring data streams. However, it is clear that the quality of detected possible correlations heavily depends upon the quality of detected anomalies.

For future work it is essential to evaluate the approach on real world data. Besides this, the prototype can be extended in various ways with the most interesting being the test of different anomaly detection algorithms for different monitoring stream data.

Acknowledgments. This work is part of TUM Living Lab Connected Mobility (TUM LLCM) project and has been funded by the Bayerisches Staatsministerium für Wirtschaft und Medien, Energie und Technologie (StMWi). We also thank our reviewers for their valuable feedback and constructive reviews.

References

1. P. Hershey and C. B. Silio. Systems of systems approach for monitoring and response across net-centric enterprise systems. In *Systems Conference, 2010 4th Annual IEEE*, pages 1–6, April 2010.
2. TUM LLCM. Integrated monitoring. `http://tum-llcm.de/project/ap3/tp32/`, 2016 (accessed February 3, 2016).
3. TUM LLCM. Visual service-management control panel. `http://tum-llcm.de/project/ap3/tp33/`, 2016 (accessed February 3, 2016).
4. TUM LLCM. Living lab connected mobility. `http://www.tum-llcm.de`, 2016 (accessed February 3, 2016).

5. Linh Thao Ly, Fabrizio Maria Maggi, Marco Montali, Stefanie Rinderle-Ma, and Wil van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Information Systems*, 54:209–234, 2015.
6. D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, Jan 1997.
7. N. Delgado, A. Q. Gates, and S. Roach. A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE Transactions on Software Engineering*, 30(12):859–872, Dec 2004.
8. Serafeim Zanikolas and Rizos Sakellariou. A taxonomy of grid monitoring systems. *Future Gener. Comput. Syst.*, 21(1):163–188, January 2005.
9. Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Survey cloud monitoring: A survey. *Comput. Netw.*, 57(9):2093–2115, June 2013.
10. Chrysostomos Zeginis, Kyriakos Kritikos, Panagiotis Garefalakis, Konstantina Konsolaki, Kostas Magoutis, and Dimitris Plexousakis. *Service-Oriented and Cloud Computing: Second European Conference, ESOCC 2013, Málaga, Spain, September 11-13, 2013. Proceedings*, chapter Towards Cross-Layer Monitoring of Multi-Cloud Service-Based Applications, pages 188–195. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
11. Michael Vierhauser, Rick Rabiser, Paul Grnbacher, Klaus Seyerlehner, Stefan Wallner, and Helmut Zeisel. Reminds: A flexible runtime monitoring framework for systems of systems. *Journal of Systems and Software*, 2015.
12. Luciano Baresi and Sam Guinea. Event-based multi-level service monitoring. In *ICWS*, pages 83–90. IEEE Computer Society, 2013.
13. Adrian Mos, Carlos Pedrinaci, Guillermo Alvaro Rey, José Manuel Gómez, Dong Liu, Guillaume Vaudaux-Ruth, and Samuel Quaireau. Multi-level monitoring and analysis of web-scale service based applications. In *Service-Oriented Computing. ICSOC/ServiceWave 2009 Workshops - International Workshops, ICSOC/ServiceWave 2009, Stockholm, Sweden, November 23-27, 2009, Revised Selected Papers*, pages 269–282, 2009.
14. D. Hawkins. Identification of Outliers. *Monographs on Statistics and Applied Probability*. Springer Netherlands, 1980.
15. P. J. Rousseeuw and A. M. Leroy. Robust Regression and Outlier Detection. John Wiley and Sons, USA, New York, 1987.
16. Bovas Abraham and Alice Chuang. Outlier Detection and Time Series Modeling. *Technometrics*, 31(2):241–248, 1989.
17. Bovas Abraham and George E. P. Box. Bayesian analysis of some outlier problems in time series. *Biometrika*, 66(2):229–236, 1979.
18. Pedro Galeano, D. Peña, and R. Tsay. Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, 101(474):654–669, 2006.
19. A. Zeevi, R. Meir, and R. Adler. Time series prediction using mixtures of experts. *Advances in Neural Information Processing*, 9:309–315, 1997.
20. A. M. Bianco, M. García Ben Ben, E. J. Martínez, and V. J. Yohai. Outlier detection in regression models with ARIMA errors using robust estimates. *Journal of Forecasting*, 20(8):565–579, 2001.
21. Ruey S. Tsay. Outliers, level shifts, and variance changes in time series. *Journal of Forecasting*, 7(1):1–20, 1988.
22. Charu C. Aggarwal. Outlier Analysis. Springer-Verlag New York, 1 edition, 2013.
23. Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41:1–58, September 2009.

24. V. J. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *Artifical Intelligence Review*, 22(2):85–126, 2004.
25. Manish Gupta, Jing Gao, Charu Aggarwal, and Jiawei Han. Outlier Detection for Temporal Data : A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 26(9):2250–2267, 2014.
26. Stephen Ranshous, Shitian Shen, Danai Koutra, Steve Harenberg, Christos Faloutsos, and Nagiza F. Samatova. Anomaly detection in dynamic networks: a survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, pages 1–27, 2015.
27. Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: A survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.
28. Ruoyi Jiang, Hongliang Fei, and Jun Huan. Anomaly localization for network data streams with graph joint sparse PCA. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 886–894, 2011.
29. C.S. Chao, D.L. Yang, and A.C. Liu. An automated fault diagnosis system using hierarchical reasoning and alarm correlation. *Proceedings 1999 IEEE Workshop on Internet Applications*, 9(2):183–202, 1999.
30. He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Dan Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. Argus: End-to-End Service Anomaly Detection and Localization From an ISP's Point of View. *In INFOCOM, Proceedings IEEE*, pages 3038–3042, 2012.
31. M. Steinder and A. Sethi. End-to-end service failure diagnosis using belief networks. *In IEEE/IFIP Network Operations and Management Symposium*, pages 375–390, 2002.
32. Chengwei Wang, Infantdani Rayan, Greg Eisenhauer, Karsten Schwan, Vanish Talwar, Matthew Wolf, and Chad Huneycutt. VScope: middleware for troubleshooting time-sensitive data center applications. *Middleware 2012*, 7662:121–141, 2012.
33. Chengwei Wang, Karsten Schwan, Vanish Talwar, Greg Eisenhauer, Liting Hu, and Matthew Wolf. A flexible architecture integrating monitoring and analytics for managing large-scale data centers. *In Proceedings of the 8th ACM international conference on Autonomic computing*, pages 141–150, 2011.
34. Mike Chen, Alice Zheng, Jim Lloyd, Michael Jordan, and Eric Brewer. Failure Diagnosis Using Decision Trees. *In Autonomic Computing*, pages 36–43, 2004.
35. Mohammad Maifi Hasan Khan, Hieu Khac Le, Hossein Ahmadi, Tarek F. Abdelzaher, and Jiawei Han. Dustminer: troubleshooting interactive complexity bugs in sensor networks. *In Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 99–112, 2008.
36. Myunghwan Kim, Roshan Sumbaly, and Sam Shah. Root cause detection in a service-oriented architecture. *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*, 41(1):93–104, 2013.
37. Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Databases Method for Very Large Databases. *ACM SIGMOD International Conference on Management of Data*, 25(2):103–114, 1996.
38. A. J. Fox. Outliers in time series. *Journal of the Royal Statistical Society*, 34(3):350–363, 1972.
39. Lilian Shiao-Yen Wu and J.R.M. Hosking. Reallocation Outliers in Time Series. *Journal of Royal Statistical Society*, 42(2):301–313, 1991.

40. Antoine Dutot, Frédéric Guinand, Damien Olivier, and Yoann Pigné. GraphStream: A Tool for bridging the gap between Complex Systems and Dynamic Graphs. In *Emergent Properties in Natural and Artificial Complex Systems. Satellite Conference within the 4th European Conference on Complex Systems (ECCS'2007)*, Dresden, Germany, October 2007.
41. Pentaho Corporation. Pentaho — data integration, business analytics and big data leaders. `http://www.pentaho.com/`, 2016 (accessed February 15, 2016).
42. Red Hat. Drools - business rules management system. `http://www.drools.org/`, 2016 (accessed February 15, 2016).