Towards a Multi-Layer IT Infrastructure Monitoring Approach based on Enterprise Architecture Information

Martin Kleehaus, Ömer Uludağ and Florian Matthes[1]

Abstract: Enterprise Architecture Management (EAM) tools play an important supporting role in IT management of organizations to align their IT infrastructure to actual business needs. The continuously measurement and observation of each layer in an enterprise architecture is critical in order to achieve an holistic view about the EA operation. This paper describes a concept how to exploit and extend the metainformation of an IT architecture documented in an EA tool in order to support a multi-layer monitoring approach that provides traceability and correlation between monitoring data extracted from several abstraction layers.

Keywords: Business Process Monitoring, Application Performance Monitoring, Infrastructure Monitoring, Log Mining, Enterprise Architecture Management

1 Introduction

The measurement and control of IT and business services delivered by an Enterprise Architecture (EA) is based on a continuous process of monitoring the instances, reporting on failures, learning from their behavior and planing subsequent actions. These steps are fundamental as, although this monitoring process takes place during service operation, they provide important information about the EA which in turn can assist for improving the alignment of the IT and business strategy.

Many well known EA frameworks emerged in the last decades like TOGAF [Ha11], ArchiMate [Gr16], ITIL [Of11], etc., that deliver a standard how to model the EA based on abstraction layers: 1) the IT infrastructure encompasses all technological aspects, 2) the application layer defines the software running in the IT infrastructure and 3) the business processes that operate on top of the aforementioned layers. Although a plethora of monitoring solutions [Kl16] have been developed to account for these layered architectures, most of these solutions specialize on a specific layer or requirement, like Business Process Monitoring [ADO00], Application Performance Monitoring [Ra12], Infrastructure Monitoring [Jo07], or Log Mining [AGL98] solutions. This makes it challenging to obtain an integrated and holistic view on the behavior and status of the EA as most tools either support only a technical viewpoint, or a business oriented viewpoint [BGP11].

In the following sections, we present an conceptual implementation how to design an integrated real-time multi-layer EA monitoring solution that establishes a link between the

[1] Technische Universtität München, Software Engineering for Business Information Systems (sebis), Boltzmannstrasse 3, 85748 Garching bei München, {martin.kleehaus, oemer.uludag, matthes}@tum.de

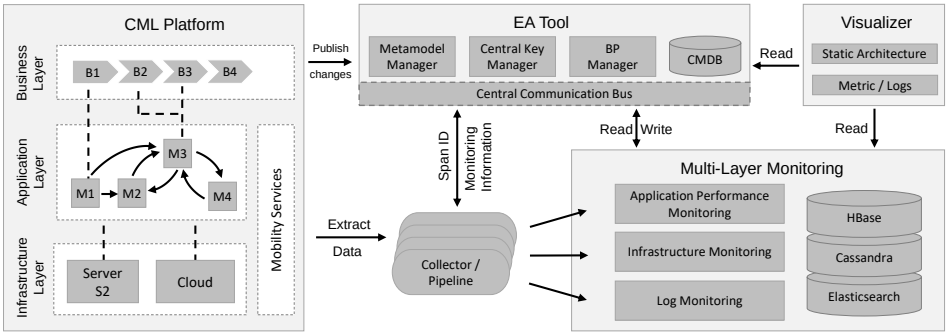

Fig. 1: Architecture of a multi-layer monitoring solution based on EA information

software and hardware components like the communication behavior, the dependencies between each components and the business processes that are supported by the services.

2 Connected Mobility Lab

The reference architecture that we use for evaluating our solution in future work will be the academic project *Connected Mobility Lab (CML)*[3]. The architecture is illustrated on the left side of figure 1. CML presents an open, digital mobility platform and is aligned on a microservice architecture. It consists of core services and mobility services. Core services undertake a supporting role by providing interfaces and required backend features. Mobility services provide business value to the end user and valuable data to other services that consume this data in order to enrich their own service.

The following use cases need to be addressed by the monitoring solution: 1) What is a root cause of an occurred error and who is accountable? 2) Which services suffer from bad performance? 3) Do the services comply with the specified Quality of Service (QoS) agreements? 4) What services fulfill a specific business process?

3 Multi-Layer Monitoring System

In the context of the proposed multi-layer monitoring solution we establish an *EA tool* enhanced with a *Configuration Management Database (CMDB)*, as it is illustrated in the top of figure 1. The tool integrates a *metamodel manager* that applies an EA model as the design-time model that defines visual representations of all monitored components like business processes, software systems, hardware elements and their relations. The metainformation of the components (id, name, type, etc.), their relations (id, communication type, etc.) and monitoring specific information like the path to log files are stored in the CMDB.

[3] http://tum-llcm.de/

Besides the EA tool which is the central control center of the monitoring solution we deploy a combination of several monitoring applications that collect data from each EA layer (right side of figure 1). However, as these applications perform their task rather self-sufficient and are connected to several database technologies (HBase, Cassandra, Elasticsearch as an example), we enhance the EA tool with a *central communication bus* that support the heterogeneous monitoring infrastructure. That means, each monitoring application has to communicate with the EA tool for extracting required information, like the component id in order to achieve a correlation between the monitored components.

Furthermore, the EA tool also includes an interface for receiving continuous deployments made on the CML application in order to keep the architecture information up to date. In particular, the introduction or the withdrawal of software components are monitored. The following sections describe the monitoring solutions in more detail.

3.1 Monitor the infrastructure layer

The infrastructure layer of the CML platform is monitored by deploying agents on the servers extracting status information about traffic, bandwith, CPU utilization, etc. Open source solutions like Nagios[4] or Sensu[5] fit for this purpose. However, these solutions do not provide information about what specific user transaction is accountable for a huge resource utilization or which application causes abnormal behavior.

One approach in order to address this challenge, we suggest to leverage the linked information in the EA tool in order to correlate hardware metrics to running transactions by using the written timestamps as a connection key. In addition, log events written by the infrastructure and the application layer can uncover further important information about the behavior of these systems. Fur this purpose, we deploy the ELK stack[6] for processing and indexing the log files. The information which IT component creates the log events and what user transaction is currently in process can be retrieved from the EA tool, passed to the Logstash pipeline and stored in Elasticsearch.

3.2 Monitor the application layer

The CML platform consist of microservices (illustrated as M_X in figure 1) which provide a bulk of different backend and mobility services. These applications are constructed from collection of software modules that were developed by different teams, in different programming languages. A huge challenge for monitoring this configuration is to identify how and to what extent the microservices are communicating with each other in order to understand system behavior and reasons about performance issues.

In order to achieve this goal we apply the application performance monitoring (APM) approach ”Dapper” described by Google [Si10] and adapted by several academic projects

[4] https://www.nagios.org/
[5] https://sensuapp.org/
[6] https://www.elastic.co/

like kieker [vHWH12] and pivottracing [MRF15], and open source projects like pinpoint[7], or zipkin[8]. Dapper provides a solution for analyzing the overall structure of a system and how components within them are interconnected by tracing transactions across microservices without changing the application code. Each transaction contains a collection of span identifier (span id) that refer to a specific Remote Procedure Call (RPC). However, as the span ids are generated from scratch by default, the APM solution has to be modified in the way that the keys describing the specific component of the platform architecture are issued by the *central key manager* from the EA tool. Furthermore, it has to be assured, that this modification has no significant performance impact on the services.

In addition, the span identifier need to be assigned to log events which are written from the particular service. This can be realised by altering the Logstash configuration file.

3.3 Monitor the business process layer

The goal of business process monitoring is to extract business events that refer to a well-defined step in the business process activity (marked as B_X in figure 1) from transaction logs. Hereby, it is challenging to know who performs the activity, what transaction events compose a whole activity and in particular when an activity has been started and finished [Do05].

To address this challenge, we propose to extend the EA tool with a *business process manager* that assists to define and manage business process events based on the trace information provided by the APM solution. Hereby, each relevant RPC refers to a business event and is mapped to one or more specific business activities that, in turn, compose a particular business case. However, in the first instance, the table for mapping RPC calls to predefined business process activities has to be done manually and kept always up to date.

4 Conclusion

In this short paper we presented an approach to enable real-time monitoring and visualizing of multi-layer Enterprise Architectures. We highlighted the concept of an EA tool with a central communication bus that supports the exchange of design-time and run-time information. Hereby, we were able to correlate monitoring data across the EA layers and assign them to each managed IT component and business process.

As this proposed concept is still in the design phase, our future work will be the elaboration of this solution. As the main research methodology, we will apply design science. First of all, we will investigate which monitoring application fits best for our needs. Afterwards we will model the details of our approach and develop a prototype accordingly. As the central EA tool we will use the academic project SocioCortex [MN11] and extend it with the above mentioned requirements. Finally, we will evaluate our prototype on CML.

[7] https://github.com/naver/pinpoint
[8] http://zipkin.io/

5 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

[ADO00] Aalst, Wil M. P. van der; Desel, Jörg; Oberweis, Andreas, eds. Business Process Management, Models, Techniques, and Empirical Studies, London, UK, UK, 2000. Springer-Verlag.

[AGL98] Agrawal, Rakesh; Gunopulos, Dimitrios; Leymann, Frank: Mining Process Models from Workflow Logs. In: Proceedings of the 6th International Conference on Extending Database Technology: Advances in Database Technology. EDBT '98, Springer-Verlag, London, UK, UK, pp. 469–483, 1998.

[BGP11] Brückmann, Tobias; Gruhn, Volker; Pfeiffer, Max: Towards Real-time Monitoring and Controlling of Enterprise Architectures Using Business Software Control Centers. In: Proceedings of the 5th European Conference on Software Architecture. ECSA'11, Springer-Verlag, Berlin, Heidelberg, pp. 287–294, 2011.

[Do05] van Dongen, B.; de Medeiros, A. K. A.; Verbeek, H. M. W.; Weijters, A. J. M. M.; van der Aalst, W. M. P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Applications and Theory of Petri Nets 2005: 26th International Conference, ICATPN 2005, Miami, USA, June 20-25, 2005. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 444–454, 2005.

[Gr16] Group, The Open: ArchiMate 3.0 Specification. Van Haren Publishing, 2016.

[Ha11] Haren, Van: TOGAF Version 9.1. Van Haren Publishing, 10th edition, 2011.

[Jo07] Josephsen, David: Building a Monitoring Infrastructure with Nagios. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.

[Kl16] Kleehaus, Martin; Landthaler, Jörg; Huth, Dominik; Matthes, Florian: Multi-Layer Monitoring and Visualization. In: Digital Mobility Platforms and Ecosystems. pp. 90–110, 2016.

[MN11] Matthes, Florian; Neubert, Christian: Wiki4EAM: Using Hybrid Wikis for Enterprise Architecture Management. In: Proceedings of the 7th International Symposium on Wikis and Open Collaboration. WikiSym '11, ACM, New York, NY, USA, pp. 226–226, 2011.

[MRF15] Mace, Jonathan; Roelke, Ryan; Fonseca, Rodrigo: Pivot tracing: dynamic causal monitoring for distributed systems. In: Proceedings of the 25th Symposium on Operating Systems Principles. ACM, pp. 378–393, 2015.

[Of11] Office, Cabinet: ITIL Service Operation 2011 Edition. The Stationery Office, Norwich, 2011.

[Ra12] Rabl, Tilmann; Gómez-Villamor, Sergio; Sadoghi, Mohammad; Muntés-Mulero, Victor; Jacobsen, Hans-Arno; Mankovskii, Serge: Solving Big Data Challenges for Enterprise Application Performance Management. Proc. VLDB Endow., 5(12):1724–1735, August 2012.

[Si10] Sigelman, Benjamin H.; Barroso, Luiz Andr; Burrows, Mike; Stephenson, Pat; Plakal, Manoj; Beaver, Donald; Jaspan, Saul; Shanbhag, Chandan: Dapper, a Large-Scale Distributed Systems Tracing Infrastructure. Technical report, Google, Inc., 2010.

[vHWH12] van Hoorn, André; Waller, Jan; Hasselbring, Wilhelm: Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In: Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering. ICPE '12, ACM, New York, NY, USA, pp. 247–248, 2012.