Towards Audio-Visual Cues for Cloud Infrastructure Monitoring

David Bermbach, Jacob Eberhardt
Information Systems Engineering Research Group
TU Berlin, Berlin, Germany
Email: {db,je}@ise.tu-berlin.de

***Abstract*—When monitoring their systems' states, DevOps engineers and operations teams alike, today, have to choose whether they want to dedicate their full attention to a visual dashboard showing monitoring results or whether they want to rely on threshold- or algorithm-based alarms which always come with false positive and false negative signals.**

In this work, we propose an alternative approach which translates a stream of cloud monitoring data into a continuous, normalized stream of score changes. Based on the score level, we propose to gradually change environment factors, e.g., music output or ambient lighting. We do this with the goal of enabling developers to subconsciously become aware of changes in monitoring data while dedicating their full attention to their primary task.

I. Introduction

Modern cloud-based enterprises increasingly follow the *you build it, you run it* paradigm where small teams of software engineers no longer only develop an application and then hand it over to other teams for testing and operation. Instead, the development team is also responsible for running the application, i.e., deploying and maintaining the system, so that their responsibility shifts from simply providing a piece of code to providing a system with strict SLAs. This is typically referred to as DevOps [1].

While this has many advantages, it also confronts developers with tasks that traditionally never were theirs to do. A good example for this is monitoring: While a traditional enterprise may have a dedicated operations team for closely observing and managing application state, this suddenly becomes a side task for application developers – a burden they are ill equipped to handle. Automation can compensate for some parts of this: threshold- or machine learning-based approaches can take automatic action to resolve issues or notify developers through alarms. Still, automatic action cannot fully replace human oversight and alarms are inherently limited by their binary state – alarm on or off – leading to a trade-off between false positive and false negative alarm states.

In this paper, we propose MultiSense, an approach that leverages the ability of the human subconscious to detect deviations from a "normal" state. For this purpose, we use monitoring results to control various aspects of the developers' environment, thus, enabling them to subconsciously become aware of faulty system states, e.g., through color changes in the ambient lighting. In contrast to the traditional pager approach, these environment factors can typically be changed gradually so that, for lack of a binary decision, false positives or false negatives become a thing of the past. Furthermore, the likelihood of a signal moving from the subconscious to a state of awareness highly depends on the intensity and regularity of the signal as well as the person's current level of concentration [2], i.e., developers will in periods of full concentration only become aware of critical system states whereas they will in periods of low concentration also become aware of smaller issues. For the evaluation of MultiSense, we are actively working on our prototype *AudioCues*, a system that is able to translate monitoring data into gradual changes in ambient background music.

This paper is structured as follows: In section II, we will discuss basic literature and related work on leveraging the subconscious for presenting information to users. Then, in section III, we will give a brief overview of MultiSense before concluding in section IV.

II. Background and Related Work

There are two cloud monitoring options: direct monitoring, where a DevOps engineer has to focus on monitoring, or peripheral monitoring where engineers focus on another task but use their subconscious to detect changes so that their focus can shift back to the monitoring task when necessary [2].

Current state-of-the-art dashboards either use direct monitoring or are based on binary alarms, even though peripheral displays have been determined effective and more efficient in the past [3].

To our knowledge, there is no broad platform comparable to MultiSense – neither as architectural concept and framework as in MultiSense nor as a prototypical implementation. However, there is some work on peripheral visual displays which have been proposed to enable monitoring for a particular source of information while working on a primary task. These displays naturally lend themselves to use as part of MultiSense. Examples of such displays are Live Wire, Waterland, or Pinwheels [4], [5].

Existing work that uses audio outputs, e.g., [3], [6] and is, thus, comparable to our AudioCues prototype is inherently based on discrete immutable audio events whereas our focus is on creating a stream of continuously changing surroundings – either through music as in the case of our prototype or

through aspects like room temperature or ambient lighting in the broader MultiSense.

III. MultiSense

In this section, we will give an overview of MultiSense and discuss environmental parameters that can be controlled.

A. Architecture and Components

On a high level, the MultiSense architecture follows a sensor-actuator model and can be divided into three parts. Figure 1 gives an overview of these parts:

The first part, the left section of the figure, comprises the *metric producers*. Here, we can use any monitoring system, e.g., Ganglia or Amazon CloudWatch. Any system that offers monitoring data, either via pull or push mechanisms, can be plugged in through an adapter mechanism.

Within the core components of MultiSense, the second part, *metric consumers* periodically poll their metric producer adapters for recent monitoring data from the underlying systems and transform this stream of data points into a stream of standard monitoring events. The events of this stream are passed on to independent *metric monitors* which serve as a kind of information hub. For each registered metric, e.g., the CPU utilization of a particular virtual machine, they have a single registered *metric analyzer* component which normalizes the stream of monitoring events into a standard score – we have used a value range of zero (normal state) to a hundred (highly critical state). Based on this, the metric monitors publish raw (score) values as well as different aggregates to a pub/sub system.

The third part, the *control targets*, comprise physical and software systems, which are able to affect the developer's environment on a continuous scale, as well as the necessary control software which registers for topics of the pub/sub system and sends control commands to their underlying control target based on received scores. For instance, a controller may decide to increase the speed of the ceiling fan upon receiving two different scores.

B. Potential Control Targets

All control targets share some basic commonalities: they do not have a binary state (in fact, a continuum is preferable) and can be controlled through electronic steering commands. Furthermore, they all affect a developer's environment, i.e., the way he or she feels due to impressions on different senses.

For example, many people like to listen to music while they work in a highly concentrated way. To gradually raise awareness, we could adjust overall playback volume or equalizer settings, i.e., per-frequency volumes, based on our normalized scores. Another option would be to change brightness and color of ambient lights or even to affect temperature, air flow and humidity through modern smart home appliances. There are virtually no limitations to which devices can be used.

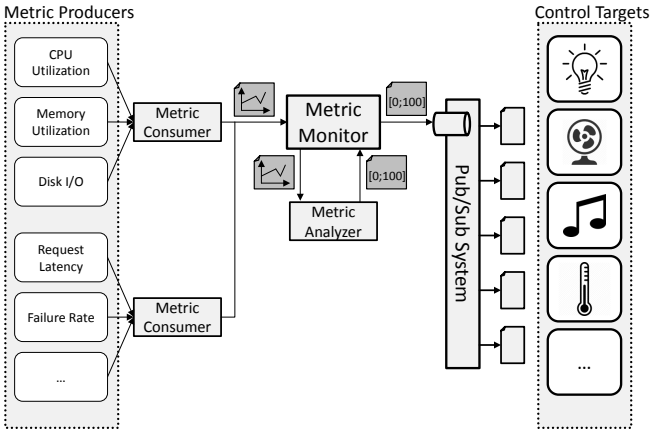

Figure 1. High-Level Architecture of MultiSense

IV. Conclusion

In this paper, we have proposed a framework and approach that leverages the ability of the human subconscious to detect deviations from a "normal" state. To reach this goal, we use cloud monitoring data to control various aspects of developers' environments, thus, enabling them to subconsciously become aware of faulty system states, e.g., through color changes in the ambient lighting or dissonances in the music output. Existing approaches, in contrast, could only express binary state changes (alarms) or required developers to dedicate their full attention to observing monitoring data.

In future work, we will continue to implement MultiSense by completing and extending our AudioCues prototype to also include control functionality for various smart devices, e.g., ambient lighting or air conditioning. We also plan to carefully evaluate MultiSense and AudioCues through empirical studies and further proof-of-concept prototypes.

References

[1] L. Bass, I. Weber, and L. Zhu, *DevOps: A Software Architect's Perspective*. Addison-Wesley Professional, 2015.

[2] P. Vickers, "Sonification for process monitoring," in *The Sonification Handbook*. Logos, 2011, pp. 455–492.

[3] M. Barra, T. Cillo, A. De Santis, U. F. Petrillo, A. Negro, V. Scarano, T. Matlock, and P. P. Maglio, "Personal webmelody: Customized sonification of web servers," *Proc. of ICAD*, 2001.

[4] M. Weiser and J. S. Brown, "The coming age of calm technology," in *Beyond calculation*. Springer, 1997, pp. 75–85.

[5] A. Dahley, C. Wisneski, and H. Ishii, "Water lamp and pinwheels: ambient projection of digital information into architectural space," in *Proc. of CHI*. ACM, 1998.

[6] O. Liechti, M. Sifer, and T. Ichikawa, "A non-obtrusive user interface for increasing social awareness on the world wide web," *Personal Technologies*, vol. 3, no. 1-2, pp. 22–32, 1999.