INSTITUT FÜR INFORMATIK

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

Florian Fittkau

Bericht Nr. 1310
November 2013
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Live Trace Visualization for System and Program Comprehension in Large Software Landscapes

Florian Fittkau

Department of Computer Science
Kiel University
D-24098 Kiel
ffi@informatik.uni-kiel.de

Abstract: The number of software systems in modern enterprise architectures is constantly increasing and thus also the complexity of such software landscapes. In addition, the knowledge of the internal behavior and utilization often gets lost.

Software visualization can provide a solution to these challenges. For instance, UML or the city metaphor are established concepts. We utilize these concepts to visualize the communication and entities in a software landscape to ease system comprehension. Our ExplorViz approach visualizes the communication taking place on both the landscape level and the system level.

In this paper, we present our PhD project: live trace visualization for system and program comprehension in large software landscapes. For this purpose, our research questions and a sketch of our approach, named ExplorViz, are described. Furthermore, this paper illustrates ideas for the planned evaluation of our approach.

1 Introduction

In many enterprises the number of software systems is constantly increasing. This can be a result of changing requirements due to, e.g., changing laws or customers the company has to satisfy. In the whole, the software systems form a software landscape including up to several hundreds of software systems.

The knowledge of communication, internals, and utilization of this software landscape often gets lost over the years [Moo03], for instance, due to missing documentation. Thus, tools supporting the understanding of the software landscape become important.

We propose our ExplorViz [FWWH13] approach for live trace visualization as a solution to assisting system and program comprehension in large software landscapes. It combines the landscape level perspective with the system level perspective. One possible scenario is the discovery of the actual communication between components.

The rest of the paper is organized as follows. Section 2 describes the goals and research questions of our planned PhD thesis. Afterwards, our ExplorViz approach is presented. In Section 4, the planned evaluation is illustrated and finally the conclusions are drawn.

2 Goals and Research Questions

We envision live trace visualization as a solution to support program and system comprehension in a large software landscape. This section provides an overview of the goals and research questions of the planned PhD thesis.

G1: Live Trace Visualization for Large Software Landscapes In the planned PhD thesis, we will develop ideas to master the live trace visualization of the large amount of incoming information typically found in a software landscape. One challenge of the visualization is the search for a suitable abstraction to display these information leading to the research question (Q.1): *How to efficiently and effectively visualize the large amount of traces in a large software landscape to support system and program comprehension?* Beside the actual visualization, we focus our work on the following four subgoals.

> **G1.1: Stable Layout Basing on Runtime Information** To support system and program comprehension, the layout of the visualization should be stable (G1.1) without the need to input upfront static information. The research question (Q1.1) for this subgoal is: *Which stable layout is suitable for live trace visualization without upfront static information?*
>
> **G1.2: Time Shift Feature** The user should be able to do a detailed analysis of one situation in a live trace visualization. We envision a time shift feature (G1.2) where the user is able to pause and resume the visualization. The research question (Q1.2) for this subgoal is: *Does a time shift feature support the comprehension process of large software landscapes in live trace visualization?*
>
> **G1.3: Addressing High Performance Aspect** The third subgoal is addressing the high performance aspect (G1.3) of a live trace visualization for a large software landscape, i.e., live processing of a large amount of monitoring data. The research question (Q1.3) for this subgoal is: *How to efficiently and effectively process several millions of events per second for live trace visualization in a large software landscape?*
>
> **G1.4: Extensibility** The visualization should be extensible (G1.4) to enable other projects to visualize their data, for instance, in the context of anomaly detection. The research question (Q1.4) is: *Which elements of the visualization need to be extensible for visualizing external data in live trace visualization?*

3 The ExplorViz Approach

Our ExplorViz approach to enable live trace visualization for large software landscapes, shown in Figure 1, includes five activities (A1 to A5). Due to space constraints our approach is only sketched. For more details, especially on the planned visualization, we refer to [FWWH13]. **A1:** The existing applications in the software landscape are monitored, e.g., with Kieker [vHWH12], providing monitoring data. **A2:** Due to the huge amount of incoming monitoring data typical for large software landscapes, we use several

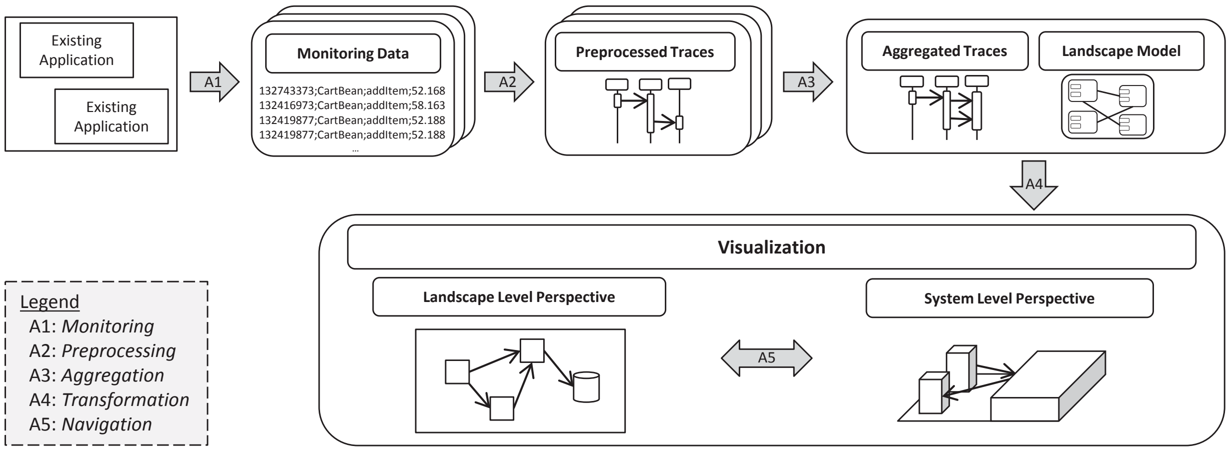

Figure 1: ExplorViz approach taken from [FWWH13]

worker nodes to preprocess the monitoring data utilizing, e.g., cloud computing. **A3:** The distributed, preprocessed traces are collected and aggregated on a single node to enable a global view of the software landscape. Additionally, this activity updates a model representation of the software landscape. **A4:** A transformation from the aggregated traces and the landscape model into a visualization model is performed in this activity. **A5:** Our live trace visualization includes a landscape level and a system level perspective.

4 Planned Evaluations

This section describes our planned evaluation. To get early qualitative feedback on our approach and visualization we will undertake a feasibility evaluation. Furthermore, we will conduct a performance evaluation to evaluate G1.3. Finally, we evaluate our live trace visualization (G1 and thus also its subgoals) by conducting a controlled experiment.

Feasibility via Case Study Our first evaluation will be a small case study for showing the feasibility of our approach by visualizing the executions in PubFlow.[1] For this purpose, we plan to ask about ten graduate students, prior unknown to our visualization, about different attributes and elements in our visualization. The case study will be conducted through a questionnaire and the results will be used to enhance our visualization.

Performance via Lab Experiments To evaluate the performance with respect to the scalability of our monitoring approach, we will conduct several lab experiments. The evaluation will measure the maximum throughput in our monitoring data processing architecture and whether our approach scales with respect to the number of monitored applications. We aim for getting data from an industrial software landscape for the evaluation. However, as a fallback solution we will use data from PubFlow.

[1]http://www.pubflow.de

Live Trace Visualization via Controlled Experiment To evaluate the goal of our live trace visualization in assisting in system and program comprehension, we will conduct a controlled experiment. Before the actual study, we will formulate hypotheses and develop an experimental design. Then, we will evaluate the design by applying it in a small scale experiment with about five participants. This reduces the risk of testing something other than our hypotheses. If major flaws are found in the design, this step might be conducted iteratively.

Afterwards, we will conduct the controlled experiment. We aim for professional software developers as participants. As a fallback solution we will conduct the evaluation with students of the course "Softwareprojekt". The number of test subjects is targeted at 80, i.e., 40 subjects in the experimental group and 40 subjects in the control group. The experiment will include comprehension and maintenance tasks [PRW04] of a software landscape. One challenge of the experiment will be the choice of a reasonable baseline for comparison with ExplorViz.

5 Conclusions

We propose live trace visualization to support system and program comprehension in large software landscapes. Therefore, we presented the goals and research questions of our planned PhD thesis. Furthermore, we sketched our approach, named ExplorViz,[2] and illustrated ideas for the planned evaluation.

References

[FWWH13] Florian Fittkau, Jan Waller, Christian Wulf, and Wilhelm Hasselbring. Live Trace Visualization for Comprehending Large Software Landscapes: The ExplorViz Approach. In *Proceedings of the 1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*. IEEE Computer Society, September 2013.

[Moo03] Leon Moonen. Exploring Software Systems. In *Proceedings of the 19th IEEE International Conference on Software Maintenance (ICSM 2003)*, pages 276–280. IEEE Computer Society, September 2003.

[PRW04] M.J. Pacione, M. Roper, and M. Wood. A Novel Software Visualisation Model to Support Software Comprehension. In *Proceedings of the 11th Working Conference on Reverse Engineering (WCRE 2004)*, pages 70–79. IEEE Computer Society, November 2004.

[vHWH12] André van Hoorn, Jan Waller, and Wilhelm Hasselbring. Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis. In *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE 2012)*, pages 247–248. ACM, April 2012.

[2]http://www.explorviz.net