Simplifying Software System Monitoring through Application Discovery with ExplorViz

Alexander Krause, Christian Zirkelbach, and Wilhelm Hasselbring
Software Engineering Group
Kiel University, Germany
{akr,czi,wha}@informatik.uni-kiel.de

Abstract

Keeping an overview of software systems is a crucial task in the area of software engineering. Often this task is supported by application performance management (APM) tools, e.g., Kieker, a monitoring framework based on dynamic software analysis. Kieker is highly extensible and customizable, but lacks a user-friendly setup configuration due to its framework characteristic. Based on this obstacle, users of our live trace visualization tool ExplorViz, which employs Kieker, experience problems during monitoring their applications.

In this paper, we report on our ExplorViz application discovery and monitoring management system (ADAMMS) to circumvent this drawback. The key concept is to utilize a software agent that simplifies the discovery of running applications within operating systems. Furthermore, the ADAMMS properly configures and manages Kieker instances to monitor these applications. Thereupon, we demonstrate how the monitoring procedure can be eased and how both open source projects – ExplorViz and Kieker mutually benefit from our approach. Finally, we conduct a first pilot study to evaluate the usability of our approach with respect to an easy-to-use application monitoring.

1 Introduction

The heterogeneity of software systems, especially in terms of programming languages, internal complexity, and communication between applications, is advancing. As a result, the development of software systems is becoming increasingly demanding. Developers and operators are forced to continuously renew their claimed knowledge of developed program internals and deployment details. New team members are even more exposed to this situation, as they must know diverse details of unknown source code and its behavior.

As a consequence, the comprehension of large software landscapes with possibly hundreds of applications is a challenging task, which requires extensive tool support. One established type of tools for this task are software software visualization (SV) tools [3, 5].

The SV tool ExplorViz[1] provides a live trace visualization of large software landscapes and their included applications [7]. For instrumentation and monitoring aspects, the software utilizes a dynamic analysis approach provided by the Kieker monitoring framework [6].

Unfortunately, users who want to setup the monitoring must manually follow a specific procedure for each desired application. This works for a small number of locally running applications, but is not suitable for enterprise software landscapes with a vast number of server nodes. Therefore, we propose an easy-to-use application monitoring approach for ExplorViz, which semi-automatically configures, instruments, and monitors (Java) applications.

To the best of our knowledge, there exist no comparable approaches in the research domain. However, we found several commercial tools with similar application discovery and monitoring management mechanisms. For example, both tools Dynatrace[2] and Instana[3] use artificial intelligence to automatically discover and monitor applications. Unfortunately, these tools are not designed to offer software visualizations, but rather show performance metrics applied to software systems and comprised applications.

The remainder of this paper is organized as follows. In Section 2, we describe the architecture and procedure of our ADAMMS. Afterwards, we present a first pilot study regarding the usability of our approach in Section 3. Finally, the conclusions are drawn and future work is revealed in Section 4.

2 The ADAMMS Approach

ExplorViz visualizes software landscapes with applications running on different servers. The ADAMMS supports this multi-server environment and enables users to manage application monitoring configurations in one graphical user interface (GUI). This differs from the previous and general approach, where users must access each server and manually start the monitoring of each application via a command line interface.

[1]https://www.explorviz.net
[2]https://www.dynatrace.com
[3]https://www.instana.com/application-management

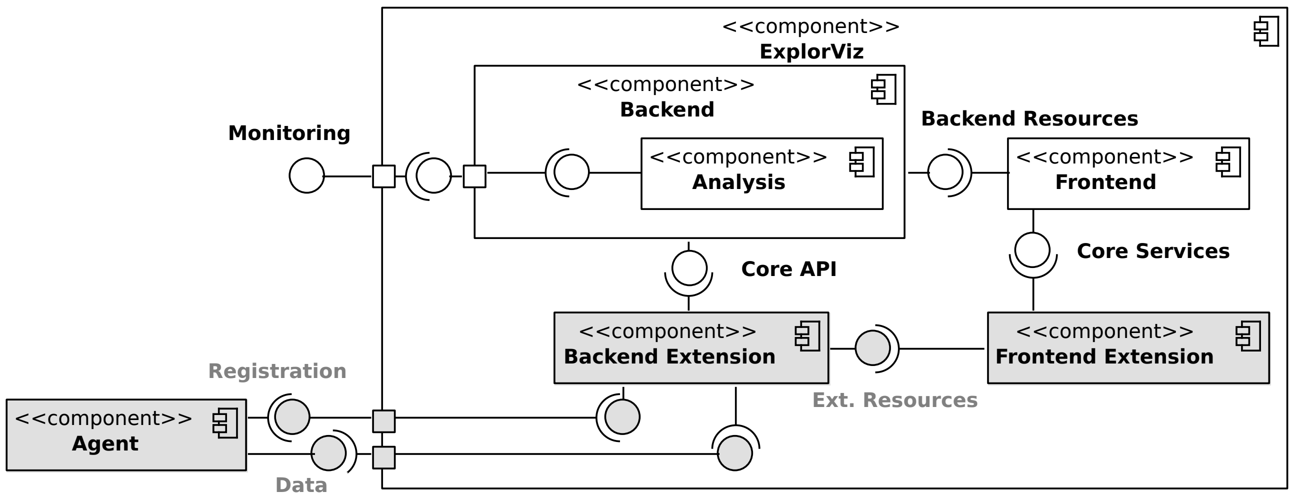

Figure 1: Architectural overview of the ExplorViz ADAMMS

2.1 Architecture

The architecture of the ExporViz ADAMMS is depicted in Figure 1. It consists of three main components (grey-colored). The ADAMMS' agent (hereinafter also referred to as agent) is a Java program that is responsible for application discovery and the monitoring configuration management. Users can deploy separate instances of this component on each server they want to scan for running applications in order to monitor them. On startup, all agents try to register at a central repository. Subsequent data is then fetched by this repository, merged, and eventually provided to clients by means of a single interface.

The extension mechanism of ExplorViz allows developers to build new components in the ExplorViz context without interfering with the core logic [8]. For that reason, the backend and frontend components provide interfaces, so that new developers can read data and use provided operations in their extension. We employ this extension mechanisms in our ADAMMS. The resulting backend extension represents the introduced central repository, whereas the frontend extension acts as client and is responsible for the visualization of data and GUI forms. The communication between agent, backend, and frontend is based on HTTP requests. The overall architecture enables a loosely coupled system, therefore easily exchangeable components.

2.2 Application Discovery and Monitoring Management

In our use case, application discovery is equal to finding running Java processes in the encompassing operating system (OS). The monitoring of these processes is started by inserting Kieker-related options to the process' execution command, e.g., the path to the Kieker Java agent, and finally restarting the process. These two mechanics are performed by implementations of our process management type (PMT) Java interface. Developers can implement new PMTs and enable the usage on different OS platforms. Both, the introduced architecture in 2.1 and the PMT interface decouple the ADAMMS' agent from our overall system and eventually enable the usage in other environments with different monitoring frameworks.

We implemented an exemplary PMT that uses the Java ProcessBuilder class to execute command line tools, such as `processes` (`ps`) or `kill`. The application discovery mechanism repetitively uses the PMTs to obtain the current OS Java process list (OSJPL). This is the first step (❶) of the application discovery pipeline as shown in Figure 2. Since processes must be restarted for the Kieker monitoring, we try to enrich the execution commands of processes with potentially found working directories and therefore absolute paths (❷). In addition to discovering processes, we also utilize so-called recognition strategies (❸). These strategies are based on a rule-based engine and try to recognize the applications behind found processes. If an application is recognized, for example by examining the execution command of its process, the strategies also recommend a suitable monitoring configuration to the user. Developers can add their custom rules to support unknown applications. Finally, the current OSJPL is merged with the data of the previous iteration (❹). Thus, we can detect and notify the user about process loss or malfunction of specific monitoring configurations.

Since we use the Kieker framework for monitoring, the management of related aspects is limited to the modification of two files (monitoring and Kieker configuration). The agent creates and hosts these files for each discovered process. Modifications, e.g., which specific source package should be monitored, and recommendations based on the recognition strategies are accessible via the respective frontend extension.

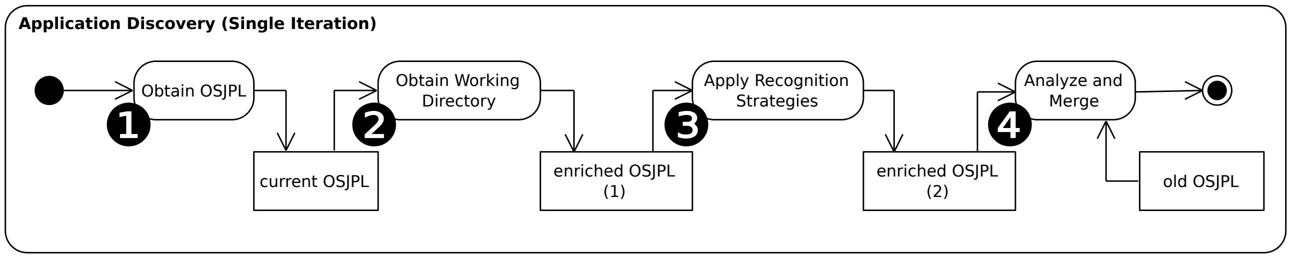

Figure 2: Application discovery and recognition procedure of the ADAMMS' agent

3 Evaluation

For our first pilot study, we examined the usability of our system focusing on the graphical user interface. Based on the results of our study, we are able to derive indications for the overall usability of our system.

We asked a researcher from our research group, who utilizes ExplorViz in his daily work, to test our ExplorViz ADAMMS. For the experiment, we designed a set of typical tasks the subject had to perform. These cover the main features of our system, i.e., detecting an application and setup of the monitoring configuration with Kieker. After the tasks were solved, we completed the experiment with an informal type of a pluralistic walkthrough, i.e., a method for usability inspection, to gather possible enhancements from the subjects' perspective [2, 4].

For the experiment, we deployed two applications (JPetStore 6 and a self-built sample application) and our agent on three different machines. One machine also hosted an ExplorViz instance. The employed software setup is provided online for repeatability.[4] The results of our first pilot study are promising. The subject was able to solve all tasks and provided valuable feedback regarding potential enhancements for the GUI. Although a usability experiment with one subject does not provide reasonable results, it still indicates the overall usability. Nevertheless, the former are required. We suggest that a second experiment with at least 30 subjects should be conducted [1]. Summarized, the experiment revealed a first indication for the necessity and usefulness of our newly developed system. Thus, the development of the ExplorViz ADAMMS should be continued in the future.

4 Conclusions and Future Work

In this paper, we reported on our application discovery and monitoring management system (ADAMMS) in ExplorViz. The approach addresses the complex and manual configuration of the Kieker framework. We explained the architecture and procedure of our ADAMMS and afterwards conducted a first pilot study in order to evaluate the usability of the system. The results show that simplifying the setup and configuration of the monitoring is a useful extension to the existing procedure. In our opinion, users are supported by our new system and thus both open source projects ExplorViz and Kieker mutually benefit from our approach. Future work includes improving the application discovery and monitoring capabilities, e.g., by machine learning methods, which analyze, learn, and eventually propose Kieker configurations for applications. Furthermore, we will enhance the support of different OS platforms by introducing new PMT interface implementations.

[4]`https://zenodo.org/record/1452411`

References

[1] J. Nielsen and T. K. Landauer. "A Mathematical Model of the Finding of Usability Problems". In: *Proceedings of CHI.* Amsterdam, The Netherlands: ACM, 1993.

[2] J. Nielsen. "Usability Inspection Methods". In: *Proccedings of the CHI.* Boston, Massachusetts, USA: ACM, 1994.

[3] S. Bassil and R. K. Keller. "Software visualization tools: survey and analysis". In: *Proceedings of the 9th International Workshop on Program Comprehension.* May 2001.

[4] A. Holzinger. "Usability Engineering Methods for Software Developers". In: *Commun. ACM* 48.1 (Jan. 2005).

[5] B. Cornelissen et al. "Trace visualization for program comprehension: A controlled experiment". In: *Proceedings of the ICPC.* May 2009.

[6] A. van Hoorn, J. Waller, and W. Hasselbring. "Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis". In: *Proceedings of ICPE.* Boston, Massachusetts, USA: ACM, 2012.

[7] F. Fittkau, A. Krause, and W. Hasselbring. "Software landscape and application visualization for system comprehension with ExplorViz". In: *Information and Software Technology* 87 (2017).

[8] C. Zirkelbach, A. Krause, and W. Hasselbring. "On the Modernization of ExplorViz towards a Microservice Architecture". In: *Combined Proceedings of the Workshops of the German Software Engineering Conference.* Ulm, Germany: CEUR Workshop Proceedings, Feb. 2018.