INSTITUT FÜR INFORMATIK

Hands-On: Experiencing Software Architecture in Virtual Reality

Christian Zirkelbach, Alexander Krause,
and Wilhelm Hasselbring

Bericht Nr. 1809
Januar 2019
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Hands-On: Experiencing Software Architecture in Virtual Reality

Christian Zirkelbach, Alexander Krause,
and Wilhelm Hasselbring

Bericht Nr. 1809
Januar 2019
ISSN 2192-6247

e-mail: {czi, akr, wha}@informatik.uni-kiel.de

Technical Report

Abstract

Recently, virtual reality (VR) and gesture-based interface devices emerged at the consumer market. Both concepts offer new visualization and interaction capabilities, which can improve the user experience when using software. In Software Engineering, exploring and comprehending software systems is often addressed via visualization techniques.

In this paper, we present our VR approach to explore software systems by using a head-mounted display and two different gesture-based interaction devices. Hence, we achieve a more immersive user experience and natural interaction, which benefits the comprehension process. Our VR approach is integrated into ExplorViz, our tool for live trace visualization of large software landscapes. In order to emphasize the advantages, we apply our combined approach on a small software system, running within our web-based tool ExplorViz. In this context, we present both, the visualization and the interaction capabilities.

1 Introduction

In the past years, virtual reality (VR) techniques emerged at the consumer market. Starting with the Oculus Rift DK1 head-mounted display (HMD), which was available at the end of 2013, the VR devices constituted a major step towards the consumer market. Based on this development, modern VR approaches became affordable and available for various research purposes. A similar development can be observed in the field of gesture-based interfaces, when Microsoft released their Kinect sensor in 2010 [1]. A combination of both techniques offers new visualization and interaction capabilities for newly created software, but can also improve reverse engineering of existing software by means of immersive user experience. In the area of software engineering, exploring and understanding software systems is often handled through 2D or 3D visualizations [2], [3]. Based on an in-depth 3D visualization and a more natural interaction, compared to a traditional 2D screen and input devices like mouse and keyboard, the user gets a more immersive experience, which benefits the comprehension process [4].

In this paper, we present a VR approach to explore software systems, visualized through the 3D software city metaphor, by using a HMD and gesture-based interaction. Our VR approach is integrated into our web-based tool ExplorViz[1] [5], [6], which offers monitoring and visualization capabilities using dynamic analysis techniques to provide a live trace visualization of large software landscapes. The tool targets system and program comprehension for developers and operators in those landscapes while providing details on the communication within an single application. We already successfully employed our software in several collaboration projects [7], [8] and experiments [9], [10].

For our VR approach, we use the HTC Vive (Vive)[2] for displaying the software city and the bundled controllers, respectively the Leap Motion sensor[3] (Leap), for the gesture recognition. We present our extended approach based on our prior work [4], where we already successfully integrated a VR feature into ExplorViz (previously with Oculus Rift DK1 and Microsoft Kinect v2). The major improvements – compared to our earlier version – affect the visualization in terms of performance and extensibility on the one hand, and on the other hand – and more significant – the gesture control with respect to

[1]https://www.explorviz.net
[2]http://www.htcvive.com
[3]http://www.leapmotion.com

ease of use and accuracy. Furthermore, the Vive features a perceptible higher display resolution, which provides a better usability, especially for reading labels within our visualization.

In the following, we describe our tool ExplorViz, explain the structure of the integrated virtual reality approach, and outline our planned demonstration. Afterwards, we illustrate our demonstration using ExplorViz in combination with a example software system, which shows the capabilities and benefits of our VR approach.

2 System Description

The system consists of our web-based tool ExplorViz (version 1.1) and our integrated VR approach. Both components will be explained in the following.

2.1 ExplorViz

Our tool consists of three major components, i.e., monitoring, analysis, and visualization. The monitoring component collects data of instrumented applications and passes the recorded data to the analysis component, where execution traces are reconstructed and aggregated. Afterwards, the corresponding traces are transformed into a data model, which constitutes the basis for the visualization in the same-named component. Finally, the visualization can be accessed through a web browser. ExplorViz features two visualization perspectives – a 2D landscape-level perspective and an application-level perspective following the 3D software city metaphor [2]. Recently, we switched our rendering engine from plain WebGL to the JavaScript library three.js[4] and restructured the system into a microservice architecture [11] for extensibility and maintainability purposes [12]. Microservices are a promising target architecture to improve the maintainability of existing, monolitic systems [13], [14].

An example application for the 3D application-level visualization is shown in Figure 1. The instrumented Java application is a simplified version of the graph database management system *Neo4j* and is running within ExplorViz. The green boxes (❶) in our 3D visualization represent packages showing their contained elements. They can be opened or closed interactively. Classes are visualized by purple boxes (❷) and the communication link is displayed by

[4] `http://www.threejs.org`

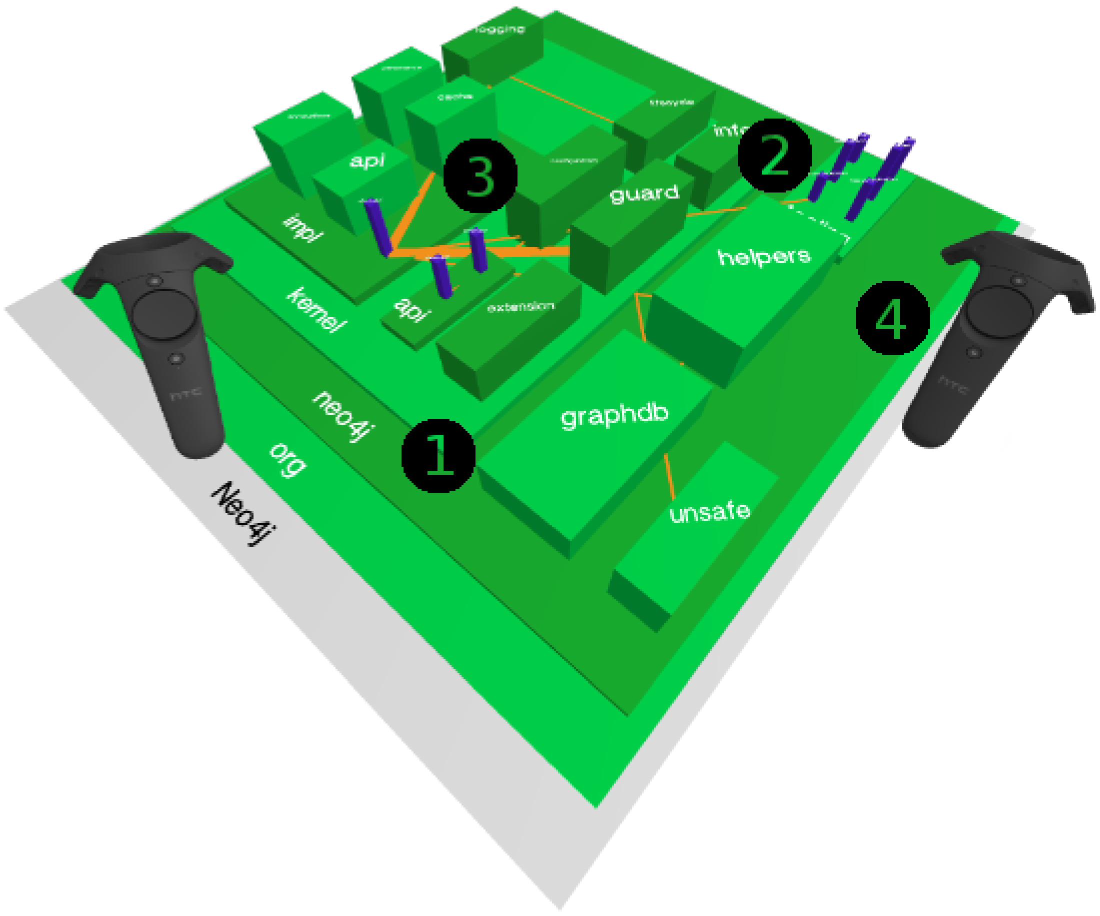


Figure 1: 3D Visualization: Simplified version of the Java application Neo4j

orange lines (❸). The width of these lines corresponds to the number of executions for a called method or operation during a specific time interval. The height of classes maps to the active instance count of objects for the respective class [6]. For visualization purposes, the figure also contains the visualized Vive controllers (❹), which will be explained later.

2.2 Integrated VR Approach

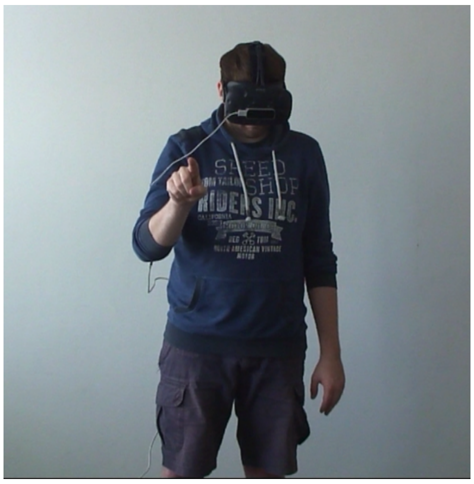
Our integrated VR approach (VR mode), builds upon our 3D visualization, which makes use of hierarchical abstractions [9], [15], in order to show details only on demand. The VR mode employs tracking abilities of HMDs to enable viewpoint rotations and introduces hand- and controller-based interactions to use intuitive motion for ExplorViz's model manipulation. A previous version of our VR mode used an Oculus Rift Development Kit 1 and a Kinect v2 sensor [4]. Our new approach uses HTC's Vive and the Leap Motion sensor for hand gesture recognition. The Leap comes with a Software Development Kit (SDK) for JavaScript and a runtime environment, which needs to be deployed on the client machine. To access the Vive and its data in our web-based ExplorViz tool, we use the JavaScript API WebVR.[5] The Vive requires a powerful computer to deliver a high and stable frame rate. Our Windows 10 system utilizes an Intel Core i5-6500 processor, a NVIDIA GeForce GTX 1070 graphics card and 16 GB of RAM. HTC bundles their HMD with two controllers and two laser emitters, called Lighthouse base stations. The latter devices are used for inside-out tracking, hence the HMD and the controllers obtain their respective positions in space. The Leap, instead, employs infrared cameras and LEDs to calculate the position of human hands in front of the device. As it is attachable to common HMDs, it allows the development of VR applications with hand-based interactions.

Based on these two different gesture-based devices, we developed custom gestures to interact with our 3D software visualization. This allows the user to interact with our visualization via four gestures – *Translation*, *Rotation*, *Selection*, and *Open Packages*.

2.3 Leap Motion Sensor

The hand-based gestures for interacting with ExplorViz's 3D model are partially based on our previous approach as presented in [4]. There was, in particular, a lack of an intuitive zooming gesture. We resolved this problem by extending the gesture for translation, which now additionally uses the

[5] `https://webvr.info`


(a) Selection (Leap)

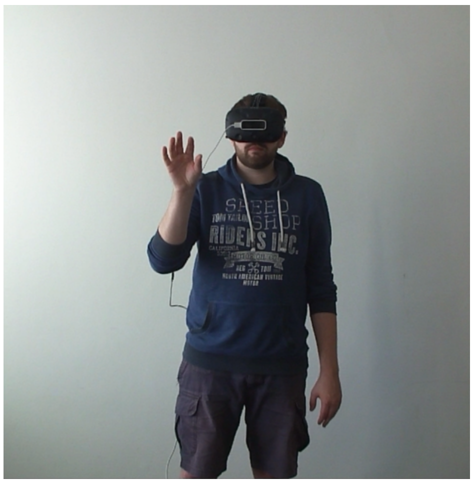

(b) Open Package (Leap)

Figure 2: VR gestures employing the Leap

z-axis for zooming. Now, to move the object freely in space, the user lifts his right hand, clenches into a fist and then starts moving the arm. The same gesture with the left hand is used for rotating the object.

Figure 2a demonstrates how packages and classes can be (de)selected for tracking communication between model elements. This discrete gesture is pre-defined in the Leap Motions JavaScript SDK and can be used out of the box. The user lifts his right hand and pokes the virtual object. This is supported by crosshairs in the middle of the user's viewpoint. The gesture triggers an event and the object behind the crosshair is (de)selected. In Figure 2b, the gesture for opening packages is illustrated. The user lifts his right hand and taps the virtual object with a finger. This activates a procedure similar to the selection gesture. Likewise, the user can perform the gesture on an open package to hide its inner structure. This gesture is pre-defined in the Leap Motion's SDK. Additionally, the hands are visualized within the HMD.

Previous tests showed, that common gesticulation in conversations leads to unintended recognition of gestures. The subsequent manipulation of the object often resulted in confusion and annoyance, hence decreasing the usability of a VR mode. To minimize this unwanted behavior we use anchor points.

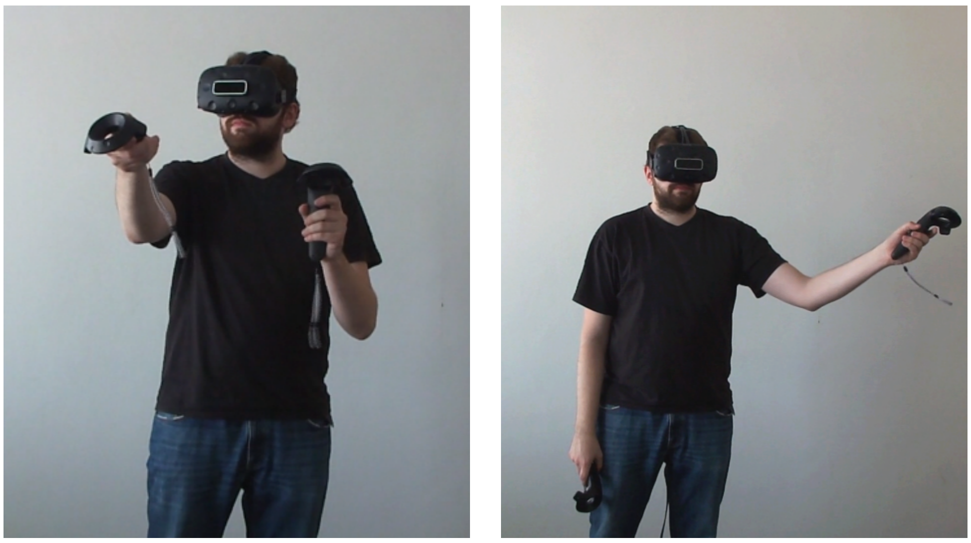

(a) Selection (Vive) (b) Translation (Vive)

Figure 3: VR gestures employing the Vive

2.4 HTC Vive Controllers

Although the Leap Motion sensor offers a very natural way of interaction with the system, its false positive rate and angle restriction reduce the usability. For this reason, we present an alternative interaction concept through the HTC Vive controllers, which are also visualized within the HMD as previously shown in Figure 1 (❹). The gesture for *selection* is presented in Figure 3a. The user aims for the intended object with a laser-like ray and presses the trackpad. A similar procedure applies for *moving* the 3D model, which is shown in Figure 3b. The user lifts the right controller, holds the bottom trigger button, and then moves the object. Further gestures involve *rotating* the model via the trackpad and *opening* and *closing* a package by pressing another button.

3 Illustration

To illustrate our implemented approach and its characteristics, we present the interactive VR mode using the previously mentioned sample software system as shown in Figure 1.

3.1 Overview

Our demo illustration showcases our extended virtual reality (VR) approach to explore software systems by using a head-mounted display (HMD) and gesture-based interaction. Therefore, we integrated this approach into our web-based tool ExplorViz, which offers monitoring and visualization capabilities using dynamic analysis techniques to provide a live trace visualization of large software landscapes. It targets system and program comprehension in those landscapes while providing details on the communication within an single application. The visualization of monitored applications follows the 3D city metaphor. We utilize HTC's Vive HMD for new capabilities, i.e., viewpoint rotation based on head-tracking, to offer an immersive user experience in those software cities.

Additionally, we apply two modern gesture-based devices for manipulating ExplorViz's 3D model. First, the Leap Motion sensor (Leap) is used for hand recognition, which allows the user to interact with the visualization. Furthermore, the sensor enables a visualization of the user's hands in our tool. This is a major improvement in comparison to our previous VR approach, which used a Kinect v2 sensor with a decoupled standalone application for gesture recognition. Second, we present an alternative interaction concept through the Vive's controllers, which are also visualized within the HMD. Finally, we illustrate the advantage over the Leap in respect of false positive rate and angle restriction, and highlight the tremendous accuracy.

3.2 Scenario

We start with a short introduction of the visualization and the related elements within our sample system, i.e., packages, classes, and communication. Afterwards, we focus on the interaction and present the gestures, which constitute the controls for the user. As we offer two different gesture recognition systems, we begin with the Leap Motion sensor. We perform the described gestures and present the intuitive interaction. Finally, we use the Vive controllers, illustrate the advantage over the Leap in respect of false positive rate and angle restriction, and highlight the accuracy. Additionally, we emphasize the benefits and the drawbacks of both gesture-based interaction concepts. All mentioned gestures, their handling, and related visualizations are showcased in an uploaded demo video. Especially the interaction of our VR approach and the differences between both employed gesture-based devices are shown. The video can been found online on YouTube, which can be accessed at `https://youtu.be/V5vluFn1dcQ`.

3.3 Goals

The intention of our demo is to reveal the advantages of utilizing alternative display and interaction concepts, i.e, based on gestures, in the area of software visualization. As an example for successful integration, we present our tool ExplorViz, which we enhanced through the VR approach. We plan to conduct experiments to validate our further developed approach and the related gestures. Additionally, as we aim for verifiability in general, we provide our open-source project ExplorViz[6] under the Apache 2.0 License. Thus, we encourage other researchers and developers to build upon our approach for their future developments. In general, we would like to see further research on the utilization of modern alternative classic input and output devices. This proposition is particularly not limited to VR, but focuses augmented reality as well.

3.4 Requirements and Setup

The demonstration needs approximately 3x3m of space for a *virtual room* (Vive). An example setup four our *virtual room* is presented in Figure 4. The requirement of space for this setup – including the tripods for positional tracking devices is illustrated. An additionally table for the computer setup, which is needed to render the visualization for the HMD, is visible in the lower left corner of the image.

[6] `https://www.explorviz.net`

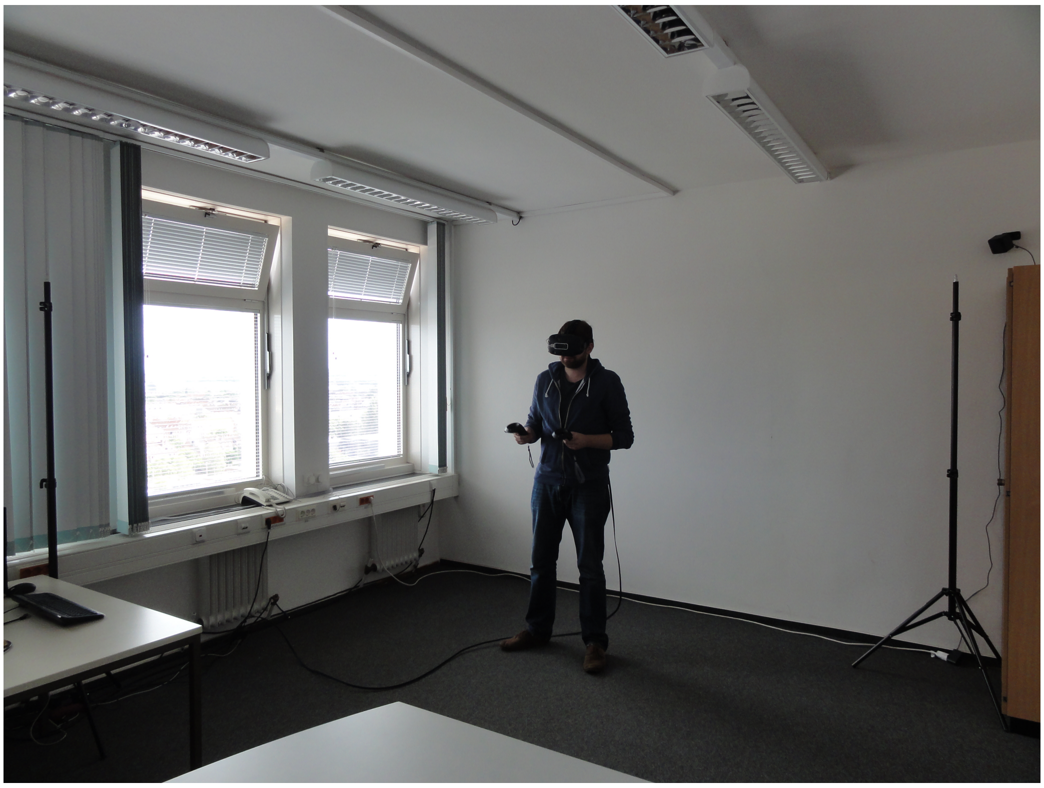


Figure 4: Setup for our virtual room

References

[1] L. Garber, "Gestural technology: Moving interfaces in a new direction," *Computer*, vol. 46, no. 10, pp. 22–25, 2013, ISSN: 0018-9162. DOI: 10.1109/MC.2013.352.

[2] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *Proceedings of the 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2007, pp. 92–99. DOI: 10.1109/VISSOF.2007.4290706.

[3] J. Waller, C. Wulf, F. Fittkau, P. Döhring, and W. Hasselbring, "SynchroVis: 3D visualization of monitoring traces in the city metaphor for analyzing concurrency," in *1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*, Sep. 2013. DOI: 10.1109/VISSOFT.2013.6650520.

[4] F. Fittkau, A. Krause, and W. Hasselbring, "Exploring software cities in virtual reality," in *Proceedings of the 3rd IEEE Working Conference on Software Visualization (VISSOFT 2015)*, 2015, pp. 130–134. DOI: 10.1109/VISSOFT.2015.7332423.

[5] F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring, "Live trace visualization for comprehending large software landscapes: The ExplorViz approach," in *1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*, Sep. 2013. DOI: 10.1109/VISSOFT.2013.6650536.

[6] F. Fittkau, A. Krause, and W. Hasselbring, "Software landscape and application visualization for system comprehension with ExplorViz," *Information and Software Technology*, 2016, ISSN: 0950-5849. DOI: 10.1016/j.infsof.2016.07.004.

[7] R. Heinrich, R. Jung, C. Zirkelbach, W. Hasselbring, and R. Reussner, "An architectural model-based approach to quality-aware devops in cloud applications," in *Software Architecture for Big Data and the Cloud*, I. Mistrik, R. Bahsoon, N. Ali, M. Heisel, and B. Maxim, Eds., Cambridge: Elsevier, Jun. 2017, pp. 69–89.

[8] R. Heinrich, C. Zirkelbach, and R. Jung, "Architectural Runtime Modeling and Visualization for Quality-Aware DevOps in Cloud Applications," in *Proceedings of the IEEE International Conference on Software Architecture Workshops (ICSAW)*, Apr. 2017, pp. 199–201. DOI: doi:10.1109/ICSAW.2017.33.

[9] F. Fittkau, A. Krause, and W. Hasselbring, “Hierarchical software landscape visualization for system comprehension: A controlled experiment,” in *Proceedings of the 3rd IEEE Working Conference on Software Visualization (VISSOFT 2015)*, IEEE, Sep. 2015, pp. 36–45.

[10] F. Fittkau, S. Finke, W. Hasselbring, and J. Waller, “Comparing Trace Visualizations for Program Comprehension through Controlled Experiments,” in *Proceedings of the 23rd IEEE International Conference on Program Comprehension (ICPC 2015)*, May 2015, pp. 266–276.

[11] W. Hasselbring and G. Steinacker, “Microservice architectures for scalability, agility and reliability in e-commerce,” in *Proceedings 2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, Gothenburg, Sweden: IEEE, Apr. 2017, pp. 243–246. DOI: `10.1109/ICSAW.2017.11`.

[12] 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 2018*, vol. Vol-2066, Ulm, Germany: CEUR Workshop Proceedings, Feb. 2018.

[13] H. Knoche and W. Hasselbring, “Using microservices for legacy software modernization,” *IEEE Software*, vol. 35, no. 3, pp. 44–49, May 2018. DOI: `10.1109/MS.2018.2141035`.

[14] ——, “Drivers and Barriers for Microservice Adoption – A Survey among Professionals in Germany,” *Enterprise Modelling and Information Systems Architectures (EMISAJ) – International Journal of Conceptual Modeling*, vol. 14, no. 1, pp. 1–35, 2019. DOI: `10.18417/emisa.14.1`.

[15] F. Fittkau, S. Roth, and W. Hasselbring, “Explorviz: Visual runtime behavior analysis of enterprise application landscapes,” in *Proceedings of the 23rd European Conference on Information Systems (ECIS 2015)*, AIS, 2015, pp. 1–13.