VR-DevOps: Visualizing and Interacting with DevOps Pipelines in Virtual Reality

Roy Oberhauser[0000-0002-7606-8226]
Computer Science Dept.
Aalen University
Aalen, Germany
e-mail: roy.oberhauser@hs-aalen.de

***Abstract*—DevOps, with its tools and practices that integrate and automate software development tasks, has become mainstream and offers a host of benefits for modern software development. While the main goal is to foster collaboration with stakeholders, the plethora of platforms and tools, the uniqueness of each pipeline, coupled with a non-uniform display of information (graphical or not), can hinder collaboration and insights, especially for non-developers. Our solution concept VR-DevOps contributes a cross-platform immersive visualization of DevOps pipelines in Virtual Reality (VR). Our prototype realization shows its feasibility, while a case-based evaluation provides insights into its capabilities.**

Keywords – DevOps; virtual reality; visualization; software engineering; continuous integration; continuous delivery; pipelines; automation workflows.

I. INTRODUCTION

DevOps [1][2] is a methodology that combines development (Dev) and operations (Ops) with automation to improve the quality and speed of software deliveries. While there is no universally agreed to definition, key principles include Continuous Integration (CI), Continuous Delivery (CD), shared ownership, workflow automation, and rapid feedback. Both the code and tool integration and automation that DevOps addresses has become indispensable to modern software development. It has been reported that 83% of developers surveyed reported being involved in DevOps-related activities [3]. Lately, Security (Sec) has often been included in DevOps, denoted at the stage where it is primarily considered, e.g., DevSecOps [4]. Despite the popularity of DevOps, there are no visualization standards for pipelines; each platform and vendor has their own, and it can thus be difficult for non-developers to grasp - and hence collaborate regarding - the current state of pipeline runs, and the processes involved in software development, testing, and delivery.

Virtual Reality (VR) offers a mediated visual digital environment created and then experienced as telepresence by the perceiver. In contrast to a 2-dimensional (2D) space, VR enables an unlimited immersive space for visualizing and analyzing models and their interrelationships simultaneously in a 3D spatial structure viewable from different perspectives. In their systematic review of the DevOps field, Khan et al. [5] identified a lack of collaboration and communication among stakeholders as the primary critical challenge. Towards addressing this collaboration challenge, our contribution leverages VR towards enabling more intuitive DevOps visualization and interaction capabilities for comprehending and analyzing DevOps pipelines, thereby supporting enhanced collaboration and communication among a larger spectrum of stakeholders. A further challenge is the finding by Giamattei et al. [6] that the landscape for DevOps tools is extremely fragmented, meaning stakeholders access various custom webpages or logs. Hence, a further goal of our solution concept is to unify the visualization and information access across heterogeneous DevOps tools.

An excerpt of our prior VR work related to Software Engineering (SE), DevOps, and workflows: VR-Git [7] and VR-GitCity [8] provide VR-based visualization of Git repositories and version control. VR-SDLC [9] (Software Development LifeCycle) uses VR to visualize lifecycle activities and artefacts in software and systems development. VR-BPMN [10] (Business Process Management Notation) portrays processes in VR. This paper contributes VR-DevOps, a solution concept for visualizing and interacting with heterogenous DevOps pipelines in VR. Our prototype realization shows its feasibility, and a case-based evaluation provides insights into its potential for DevOps pipeline comprehension, analysis, and collaboration.

This paper is organized as follows: the next section discusses related work. Section 3 presents our solution concept. Section 4 describes our realization, followed by an evaluation in Section 5. Finally, a conclusion is drawn.

II. RELATED WORK

For VR-based DevOps-related work, VIAProMa [11] provides a visual immersive analytics framework for project management. DevOpsUseXR is mentioned in the paper as an eXtended Reality (XR) approach for incorporating end users to allow them to directly provide feedback in Mixed Reality (MR) regarding their experience when using a specific MR app. In contrast, our solution concept is independent of the software type being built in the pipeline, and is purely virtual, remaining consistent, app-independent, and focusing on visualizing and collaborating with regard to the DevOps pipeline. The systematic review of DevOps tools by Giamattei et al. [6] does not mention any VR, XR, or MR tools.

Non-VR based DevOps work includes DevOpsML [12], a platform modeling language and conceptual framework for modeling and configuring DevOps engineering processes and platforms. DevOpsUse [13] expands DevOps to collaborate more closely with end users. The authors also state that there is in general a research gap in applying information visualization to software engineering data, and that this needs further investigation. This concurs with our view, as we were not able to find much VR or non-VR work related to DevOps visualization.

III. SOLUTION CONCEPT

Our VR-DevOps solution concept is shown in blue relative to our other VR solutions in Figure 1. VR-DevOps is based on our generalized VR Modeling Framework (VR-MF) (detailed in [10]). VR-MF provides a VR-based domain-independent hypermodeling framework addressing four aspects requiring special attention when modeling in VR: visualization, navigation, interaction, and data retrieval. Our VR-based solutions specific to the SE and Systems Engineering (SysE) areas include: VR-DevOps (the focus of this paper, shown in blue), VR-V&V (Verification and Validation) [14], for visualizing aspects related to quality assurance, VR-TestCoverage [15] for visualizing in VR which tests cover what test target artefacts, VR-Git [7] and VR-GitCity [8] for different ways of visualizing Git repositories in VR. In the Enterprise Architecture (EA) and Business Process (BP) space (EA & BP), we developed VR-EA [16] to support mapping EA models to VR, including both ArchiMate as well as BPMN via VR-BPMN [10]; VR-EAT [17] adds enterprise repository integration (Atlas and IT blueprint integration); VR-EA+TCK [18] adds knowledge and content integration; VR-EvoEA+BP [19] adds EA evolution and Business Process animation; while VR-ProcessMine [20] supports process mining in VR. Since DevOps (or DevSecOps or DevOpsUse) can be viewed as inter-disciplinary, at least for software organizations we view the EA and BP area as potentially applicable for VR-DevOps to support synergies, more holistic insights, and enhanced collaboration across the enterprise and organizational space.

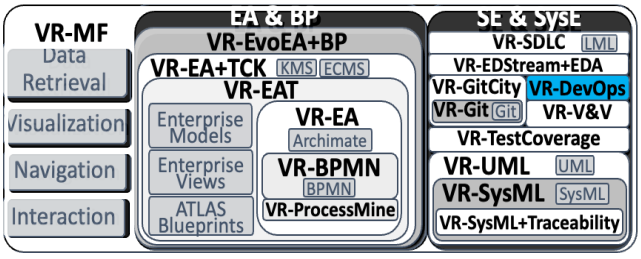

Figure 1. Conceptual map of our various published VR solution concepts with VR-DevOps highlighted in blue.

Work supporting our view that an immersive VR experience can be beneficial for analysis of software-related issues include Müller et al. [21], who compared VR vs. 2D for a software analysis task. They found no significant decrease for VR in comprehension and analysis time. While interaction time was less efficient, VR improved the user experience, was more motivating, less demanding, more inventive/innovative, and more clearly structured.

A. Visualization in VR

A horizontal *pipeline hyperplane* represents a pipeline, holding vertical semi-transparent colored boxes called *run planes* (see Figure 2), which are ordered chronologically left to right. A *run plane* represents a pipeline *run*, which is colored based on status (green=success, yellow=in progress, red=error, grey= aborted). Hyperplanes also enable inter-project pipeline differentiation for larger portfolio scenarios involving multiple pipelines. The bottom of each run plane encloses a directed graph of sequential stages (cubes) of the pipeline between a start (black sphere) and an end (black sphere), while vertically stacked smaller cubes linked with lines above each stage show the internal *steps* within a stage. A cube with black borders is used to represent the entire run, and is all that is shown when a run is collapsed (e.g., to reduce visual clutter); on its front various details are depicted (ID, run duration, circular percentage of stages with status). The visualization form remains consistent across DevOps tools.

B. Navigation in VR

Two navigation modes are incorporated in our solution: default gliding controls for fly-through VR, while teleporting instantly places the camera at a selected position via a selection on the VR-Tablet. Teleporting is potentially disconcerting, but may reduce the likelihood of VR sickness.

C. Interaction in VR

User-element interaction is supported primarily through VR controllers and a *VR-Tablet*. The VR-Tablet is used to provide detailed context-specific element information. It includes a *virtual keyboard* for text entry via laser pointer key selection. On a hyperplane corner, an *anchor sphere*

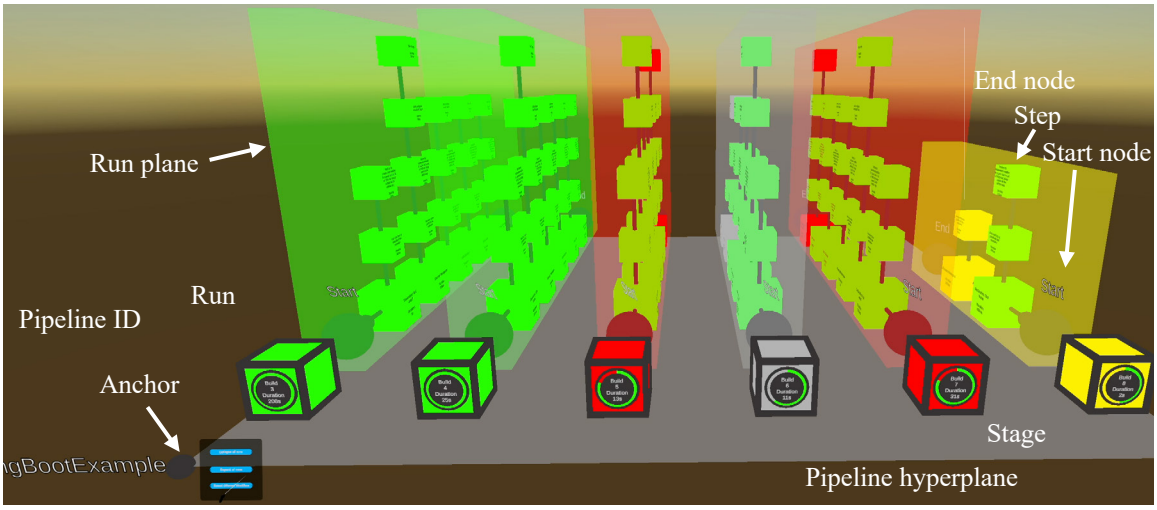

Figure 2. Hyperplane (annotated) of SprintBootExamaple Jenkins pipeline showing vertical colored run planes on a pipeline hyperplane.

affordance (labeled with its pipeline ID) supports moving, hiding (collapsing), or showing (expanding) hyperplanes, as shown in the bottom left of Figure 2.

IV. REALIZATION

In our prototype realization, VR visualization aspects were implemented using Unity. It is supported by a Data Hub implemented in Python that integrates and stores all data in JSON. All integrations with DevOps tools use their Web APIs via our tool-specific Adapters in our Data Hub. The MongoDB Atlas cloud is used for storage (easily switched to a local MongoDB). To demonstrate the tool or platform independence (i.e., heterogeneity) of our solution concept, we chose to integrate with Jenkins, exemplifying a private cloud, as well as SemaphoreCI to exemplify a public cloud tool.


```
1793     "name": "SpringBootExample",
1794     "runs": [
1795       {
1796         "id": "6",
1797         "name": "#6",
1798         "status": "ABORTED",
1799         "durationMillis": 11910,
1800         "stages": [
1801           {
1802             "id": "8",
1803             "name": "Declarative: Tool Install",
1804             "status": "SUCCESS",
1805             "durationMillis": 160,
1806             "steps": [
1807               {
1808                 "id": "9",
1809                 "name": "Use a tool from a predefined Tool Installation",
1810                 "status": "SUCCESS",
1811                 "durationMillis": 47,
1812                 "errorMessage": "",
1813                 "description": [
1814                   "maven3"
```


Figure 3. Snippet of VR-DevOps common run representation in JSON.

A CD pipeline is an automated expression of the process for preparing software for delivery. A Jenkins pipeline is a set of Jenkins plugins with a set of instructions specified in a text-based Jenkinsfile using Groovy syntax. It can be written in a scripted or declarative syntax, and typically defines the entire build process, including building, testing, and delivery. Concepts involved can include agents (executors), nodes (machines), stages (subset of tasks), and steps (a single task). A SemaphoreCI pipeline is described via a YAML syntax. We created our own common generic JSON format to store pipeline information, see Figure 3. A pipeline instance refers to a run. The refresh rates can be configured for Data Hub state retrieval from Unity and for each Adapter's Web APIs calls.

V. EVALUATION

The evaluation of our solution concept is based on the design science method and principles [22], in particular a viable artifact, problem relevance, and design evaluation (utility, quality, efficacy). A scenario-based case study is used (assuming the user may not be a developer): the *Status* scenario focuses on comprehension (run state, number of runs), while the *Analysis* scenario focuses on information retrieval (towards problem identification or resolution). To demonstrate the heterogeneity of the solution, screenshots of runs from Jenkins or SemaphoreCI are interchangeably used.

For Jenkins, the SpringBoot PetClinic example pipeline [23] includes 39 Java files and 1335 Lines of Code (LOC). For SemaphoreCI, the Android App example pipeline [24] includes 13 Kotlin files and 287 LOC. An additional step with an artificial error was inserted into the SpringBoot example for illustration purposes in a second version of the pipeline.

A. Status Scenario

Analogous to a dashboard, a VR-DevOps stakeholder should be able to readily comprehend and assess the current status and state of the various pipeline runs, exemplified for Jenkins in Figure 2 and SemaphoreCI in Figure 10. Each run may execute different steps and stages (e.g., due to an abort or error). Fully collapsed run planes provide a purely high-level overview with relevant info on the black-lined cubes, as in Figure 11. In contrast, every DevOps tool has its own web interface and way of accessing information, illustrated via screenshots of Jenkins in Figure 4 and SemaphoreCI in Figure 5. Retrieval of equivalent status and state details typically requires multiple separate web page requests, thus improving utility and efficacy, especially for increasing pipeline complexity, pipeline versions, and large scale-out of runs.

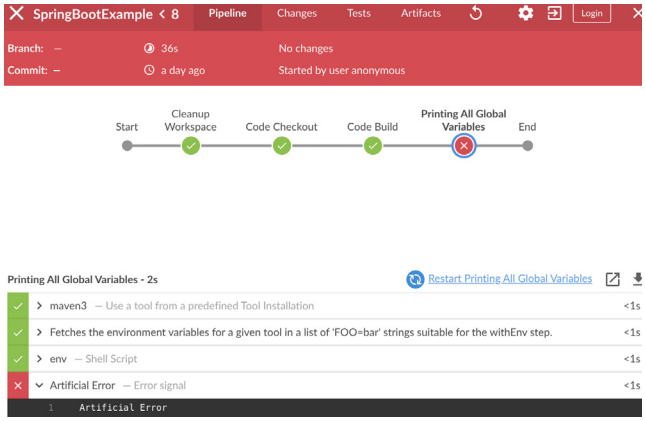

Figure 4. Jenkins tool web screenshot.

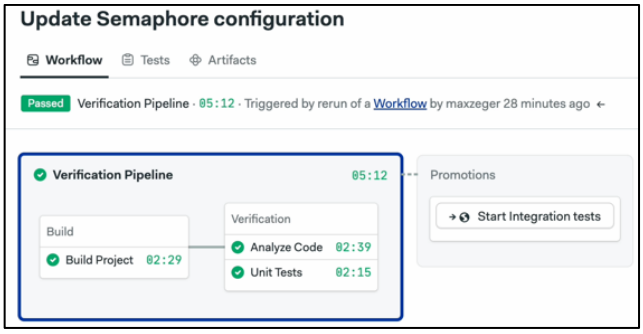

Figure 5. SemaphoreCI tool web screenshot.

B. Analysis Scenario

VR-DevOps supports issue analysis via immersive visual patterns and contrasts, visually revealing differences in pipeline versions and the detailed steps executed shown for the Android App in Figure 6. The contrasts with its YAML (YAML Ain't Markup Language) pipeline definition in Figure

8, which contains many details (difficult for all stakeholders to grasp) but lacks status info, or Figure 5, which provides simplified status, but lacks sufficient overall details. Immersive visual differentiation of runs via perspective and alignment is shown for PetClinic in Figure 7; grasping the pipeline structure from its Groovy file in Figure 9 would be more difficult for non-developers. Visual depiction and differentiation can help support the inclusion of non-tech-savvy stakeholders, improving the speed of assessments and the quality of analysis tasks by including more stakeholders. The VR-Tablet provides additional context-specific metadata and error messages about a block (Figure 12 left), step or stage task instructions (Figure 12 right), or its raw log (Figure 13 left) or pipeline info (Figure 13 right) for developers. This consolidation, in conjunction with visual differentiation, could improve the utility and efficacy of analysis tasks, especially when considering increasing pipeline complexity, pipeline versions, and large scale-out of runs.

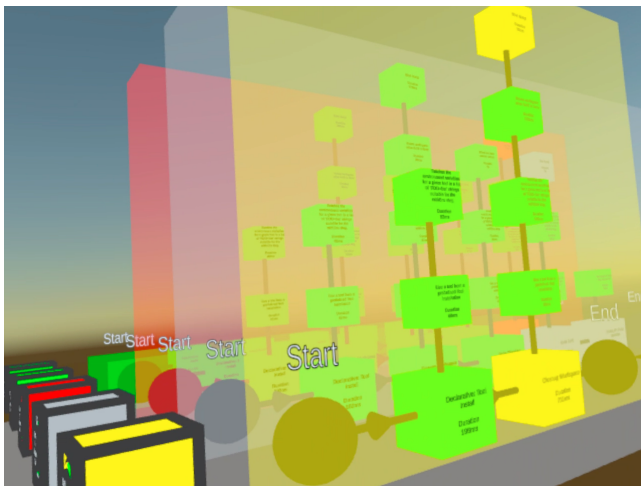

Figure 6. Immersive analysis via visual colored run comparison of stage/step status (for SemaphoreCI pipeline).

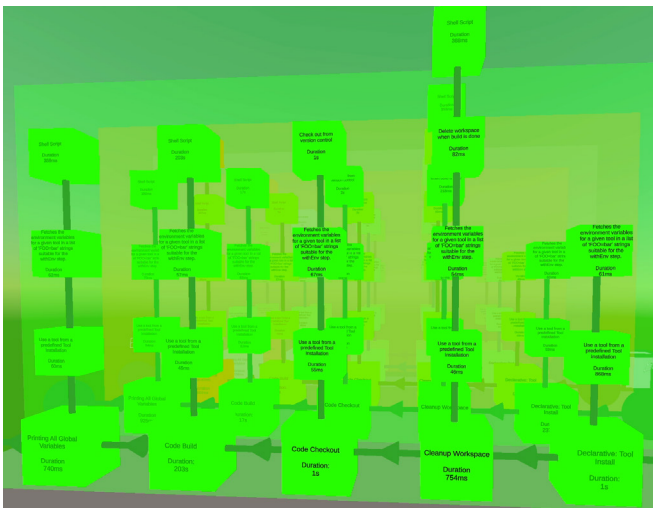

Figure 7. Immersive analysis of pipeline changes via visual alignment of stage/steps (for Jenkins pipeline).

```yaml
version: v1.0
name: Verification Pipeline
agent:
  machine:
    type: e1-standard-2
    os_image: ubuntu2004
  containers:
    - name: main
      image: 'registry.semaphoreci.com/android:29'
global_job_config:
  env_vars:
    - name: ADB_INSTALL_TIMEOUT
      value: '10'
  prologue:
    commands:
      - checkout
      - cache restore gradle-wrapper
      - cache restore gradle-cache
      - cache restore android-build
blocks:
  - name: Build
    task:
      jobs:
        - name: Build Project
          commands:
            - ./gradlew bundleDebug
      epilogue:
        on_pass:
          commands:
            - cache clear
            - cache store gradle-wrapper ~/.gradle/wrapper
            - cache store gradle-cache ~/.gradle/caches
            - cache store android-build ~/.android/build-cache
  - name: Verification
    task:
      jobs:
        - name: Analyze Code
          commands:
            - ./gradlew lint
        - name: Unit Tests
          commands:
            - ./gradlew testDebugUnitTest
      epilogue:
        always:
          commands:
            - artifact push job --expire-in 2w --destination reports/ app/build/reports/
promotions:
  - name: Start Integration tests
    pipeline_file: integration-tests.yml
```

Figure 8. SemaphoreCI Android App pipeline in YAML format

```groovy
pipeline {
    agent any
    tools {
        maven 'maven3'
    }
    options {
        buildDiscarder logRotator(
                    daysToKeepStr: '15',
                    numToKeepStr: '10'
            )
    }
    environment {
        APP_NAME = "DCUBE_APP"
        APP_ENV  = "DEV"
    }
    stages {
        stage('Cleanup Workspace') {
            steps {
                cleanWs()
                sh """
                echo "Cleaned Up Workspace for ${APP_NAME}"
                """
            }
        }
        stage('Code Checkout') {
            steps {
                checkout([
                    $class: 'GitSCM',
                    branches: [[name: '*/main']],
                    userRemoteConfigs: [[url: 'https://github.com/spring-projects/spring-petclinic.git']]
                ])
            }
        }
        stage('Code Build') {
            steps {
                sh 'mvn install -Dmaven.test.skip=true'
            }
        }
        stage('Printing All Global Variables') {
            steps {
                sh """
                env
                """
                error "Artificial Error"
```

Figure 9. SpringBoot PetClinic Jenkins pipeline in Groovy (snippet)

VI. CONCLUSION

VR-DevOps provides an immersive solution concept for visualizing, analyzing, and interacting with DevOps pipeline runs in VR. The realization prototype showed its feasibility, and the case-based evaluation showed its potential to support typical scenarios such as status and analysis. The solution concept is DevOps tool-independent, hiding the differences that the fragmented DevOps tool landscape might present to non-tech-savvy stakeholders. It thus provides a way towards broader inclusion of various DevOps stakeholders, and can thus support greater collaboration and communication to address one of the top challenges facing DevOps. Combining VR-DevOps with our other VR solutions, such as VR-Git, VR-GitCity, VR-SDLC, VR-EA, VR-V&V, VR-TestCoverage, etc., can support more holistic DevOps insights.

Future work includes: a VR-native collaboration and annotation capability, additional DevOps tool integrations, and a comprehensive industrial empirical study.

ACKNOWLEDGMENT

The authors would like to thank Maximilian Zeger for his assistance with the design, implementation, and evaluation.

REFERENCES

[1] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," in IEEE Software, vol. 33, no. 3, pp. 94-100, May-June 2016.

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

[3] L. Dodd and B. Noll, "State of CI/CD Report 2024: The Evolution of Software Delivery Performance," SlashData and the Continuous Delivery Foundation, 2024.

[4] GitLab, "A Maturing DevSecOps Landscape," 2021. [Online]. Available from: https://about.gitlab.com/images/developer-survey/gitlab-devsecops-2021-survey-results.pdf 2024.09.12

[5] M. S. Khan, A. W. Khan, F. Khan, M. A. Khan, and T. K. Whangbo, "Critical Challenges to Adopt DevOps Culture in Software Organizations: A Systematic Review," in IEEE Access, vol. 10, pp. 14339-14349, 2022.

[6] L. Giamattei et al. "Monitoring tools for DevOps and microservices: A systematic grey literature review," Journal of Systems and Software, vol. 208, 2024, p.111906.

[7] R. Oberhauser, "VR-Git: Git Repository Visualization and Immersion in Virtual Reality," 17th Int'l Conf. on Software Engineering Advances (ICSEA 2022), IARIA, 2022, pp. 9-14.

[8] R. Oberhauser, "VR-GitCity: Immersively Visualizing Git Repository Evolution Using a City Metaphor in Virtual Reality," International Journal on Advances in Software, 16 (3 & 4), 2023, pp. 141-150.

[9] R. Oberhauser, "VR-SDLC: A Context-Enhanced Life Cycle Visualization of Software-or-Systems Development in Virtual Reality," In: Business Modeling and Software Design (BMSD 2024), LNBIP, vol 523, Springer, Cham, 2024, pp. 112-129, https://doi.org/10.1007/978-3-031-64073-5_8.

[10] R. Oberhauser, C. Pogolski, and A. Matic, "VR-BPMN: Visualizing BPMN models in Virtual Reality," In: Shishkov, B. (ed.) Business Modeling and Software Design (BMSD 2018), LNBIP, vol. 319, Springer, 2018, pp. 83–97, https://doi.org/10.1007/978-3-319-94214-8_6.

[11] B. Hensen and R. Klamma, "VIAProMa: An Agile Project Management Framework for Mixed Reality," In: Augmented Reality, Virtual Reality, and Computer Graphics (AVR 2021), LNCS, vol 12980, Springer, Cham, 2021, pp. 254-272.

[12] A. Colantoni, L. Berardinelli, and M. Wimmer, "DevopsML: Towards modeling devops processes and platforms," In: 23rd ACM/IEEE Int'l Conf. Model Driven Engineering Languages and Systems: Companion Proc., ACM, 2020, pp. 1-10.

[13] I. Koren, "DevOpsUse: A Community-Oriented Methodology for Societal Software Engineering," In: Ernst Denert Award for Software Engineering 2020, Springer, 2022, pp. 143-165.

[14] R. Oberhauser, "VR-V&V: Immersive Verification and Validation Support for Traceability Exemplified with ReqIF, ArchiMate, and Test Coverage," Int'l Journal on Advances in Systems and Measurements, 16 (3 & 4), 2023, pp. 103-115.

[15] R. Oberhauser, "VR-TestCoverage: Test Coverage Visualization and Immersion in Virtual Reality," In: Proc. Fourteenth Int'l Conf. on Advances in System Testing and Validation Lifecycle (VALID 2022), IARIA, 2022, pp. 1-6.

[16] R. Oberhauser and C. Pogolski, "VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN," In: Business Modeling and Software Design (BMSD 2019), LNBIP, vol. 356, Springer, Cham, 2019, pp. 170-187, https://doi.org/10.1007/978-3-030-24854-3_11.

[17] R. Oberhauser, P. Sousa, and F. Michel, "VR-EAT: Visualization of Enterprise Architecture Tool Diagrams in Virtual Reality," In: Business Modeling and Software Design (BMSD 2020), LNBIP, vol 391, Springer, 2020, pp. 221-239. https://doi.org/10.1007/978-3-030-52306-0_14.

[18] R. Oberhauser, M. Baehre, and P. Sousa, "VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality," In: Business Modeling and Software Design (BMSD 2022), LNBIP, vol 453, Springer, 2022, pp. 122-140. https://doi.org/10.1007/978-3-031-11510-3_8.

[19] R. Oberhauser, M. Baehre, and P. Sousa, "VR-EvoEA+BP: Using Virtual Reality to Visualize Enterprise Context Dynamics Related to Enterprise Evolution and Business Processes," In: Business Modeling and Software Design (BMSD 2023), LNBIP, vol 483, Springer, 2023, pp. 110-128, https://doi.org/10.1007/978-3-031-36757-1_7.

[20] R. Oberhauser, "VR-ProcessMine: Immersive Process Mining Visualization and Analysis in Virtual Reality," Fourteenth International Conf. on Information, Process, and Knowledge Management (eKNOW 2022), IARIA, 2022, pp. 29-36.

[21] R. Müller, P. Kovacs, J. Schilbach, and D. Zeckzer, "How to master challenges in experimental evaluation of 2D versus 3D software visualizations," In: 2014 IEEE VIS International Workshop on 3Dvis (3Dvis), IEEE, 2014, pp. 33-36.

[22] A.R. Hevner, S.T. March, J. Park, and S. Ram, "Design science in information systems research," MIS Quarterly, 28(1), 2004, pp. 75-105.

[23] B. Wilson. *Jenkins Pipeline Tutorial For Beginners*. [Online]. Available from: https://devopscube.com/jenkins-pipeline-as-code/ 2024.09.12

[24] GitHub. *Semaphore demo CI/CD pipeline for Android*. [Online]. Available from: https://github.com/semaphoreci-demos/semaphore-demo-android/ 2024.09.12

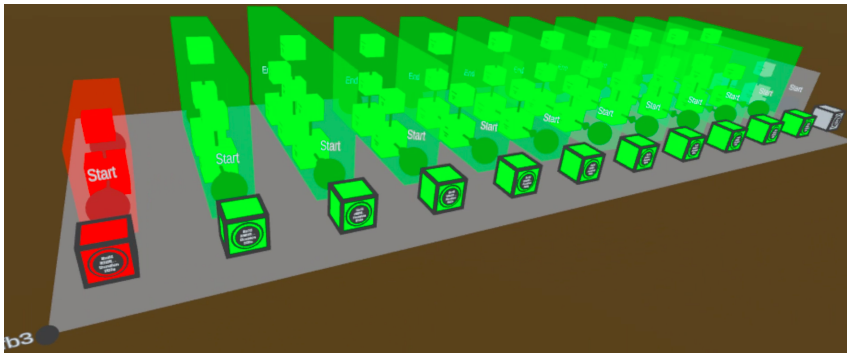

Figure 10. VR-DevOps run status for a set of SemaphoreCI pipeline runs showing expanded step details and an aborted process in grey on the far right.

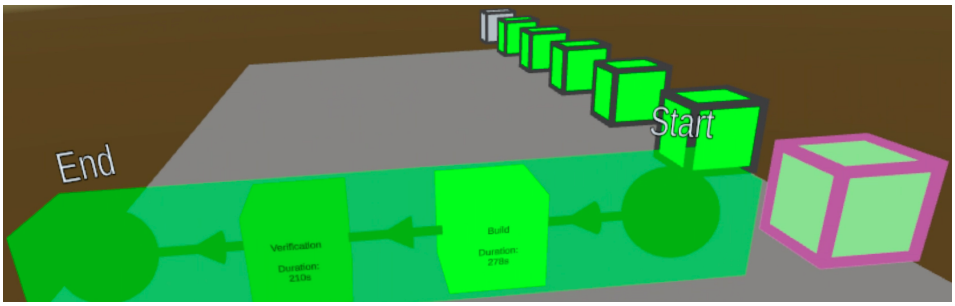

Figure 11. Collapsed SemaphoreCI runs on a pipeline hyperplane with stages expanded (and steps collapsed) for a selected run.

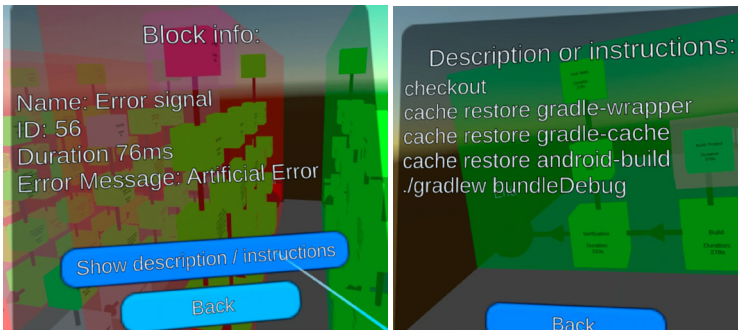

Figure 12. VR-Tablet shows contextual element details: metadata (left) and instructions/description (right).

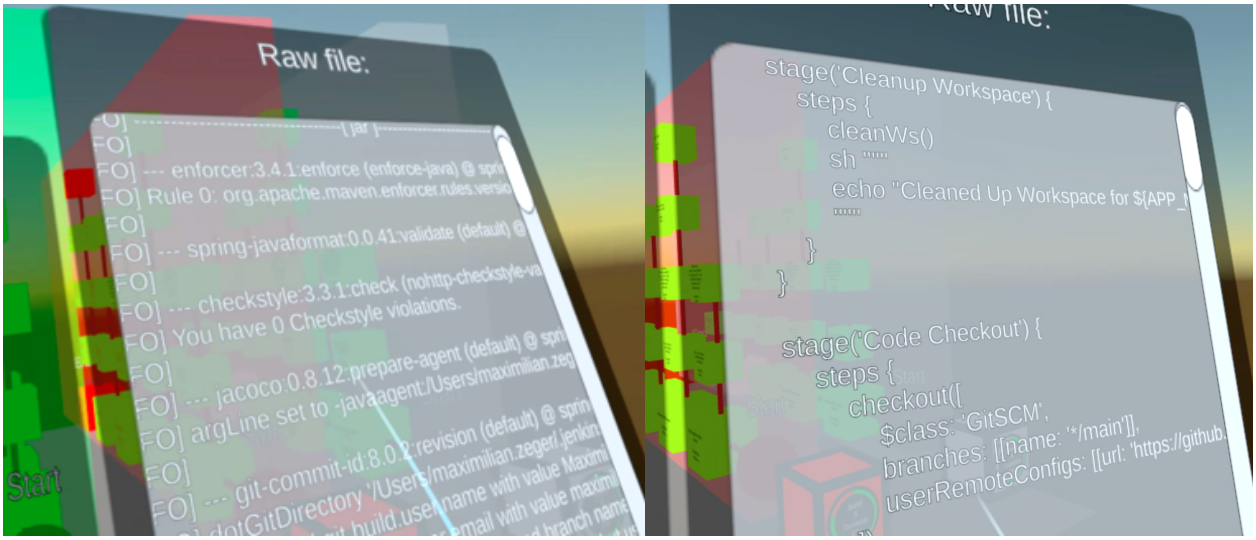

Figure 13. VR-Tablet offers raw file access to log (left) and pipeline (right).