

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Immersive visualization of CI/CD pipelines

Anete Medina Pereira



Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Rui Pedro Amaral Rodrigues

Supervisor: Prof. Daniel Filipe Martins Tavares Mendes

July 28, 2025

Immersive visualization of CI/CD pipelines

Anete Medina Pereira

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Alexandre Miguel Barbosa Valle de Carvalho

Referee: Prof. Alfredo Manuel Dos Santos Ferreira Júnior

Referee: Prof. Rui Pedro Amaral Rodrigues

July 28, 2025

Resumo

À medida que as *pipelines* de CI/CD (*Continuous Integration/ Continuous Delivery or Deployment*) se tornam cada vez mais complexas, os desenvolvedores de software têm mais dificuldade para perceber as estruturas dos *workflows*, rastrear falhas e navegar pelos históricos de execução. As ferramentas tradicionais, como o GitHub Actions, usam registros textuais e interfaces 2D que podem dificultar a visualização de relações temporais e padrões de concorrência em várias execuções de *pipelines*.

Esta dissertação explora como a Realidade Virtual (RV) pode melhorar a visualização de *workflows* CI/CD. Usando uma metáfora de uma fábrica, os dados da *pipeline* são traduzidos para ambientes imersivos 3D em que: os *workflows* tornam-se pisos, com os seus *workflow runs* correspondentes representadas como blocos colocados em cada piso, os *jobs* tornam-se sub-blocos dentro dessas *runs* e o tempo desdobra-se como profundidade espacial. Os utilizadores podem percorrer o histórico da *pipeline* e compreender os padrões de execução através de movimentos naturais.

Foi implementado um protótipo funcional em RV utilizando o Unity que transforma os dados do GitHub Actions num espaço 3D interativo. O sistema inclui algoritmos de mapeamento temporal, organização de *layout* paralelo e sequencial para *runs* e *jobs* concorrentes e dependentes, compressão de tempo para períodos inativos e áudio espacial para deteção de falhas. Os utilizadores podem explorar o histórico do *workflow*, identificar problemas e compreender o comportamento da *pipeline* através da navegação espacial.

Um estudo de utilizadores comparou o sistema em RV com a interface *web* do GitHub Actions, utilizando tarefas de navegação temporal, deteção de falhas e depuração (*debugging*). Em RV, os utilizadores concluíram as tarefas mais rapidamente e obtiveram taxas de sucesso mais elevadas do que quando utilizaram a interface do GitHub Actions. O sistema em RV recebeu pontuações de usabilidade mais altas, tendo os participantes considerado o mesmo mais fácil de usar e menos complexo.

Esta pesquisa fornece diretrizes práticas para a visualização imersiva de CI/CD. Ela demonstra que os layouts espaciais em 3D ajudam nas tarefas de visão geral e na navegação temporal, enquanto as interfaces tradicionais funcionam melhor para análises detalhadas.

Palavras-chave: análise imersiva, realidade virtual, visualização de CI/CD, engenharia de software, GitHub Actions, visualização espacial, padrões temporais, deteção de falhas

Classificação ACM:

- Human-centered computing → Human computer interaction (HCI)
→ Interaction paradigms → Virtual reality
- Human-centered computing → Visualization → Visualization application domains → Information visualization
- Software and its engineering → Software creation and management
→ Software development process management

Abstract

As CI/CD pipelines become increasingly complex, developers struggle to understand workflow structures, track failures, and navigate execution histories. Traditional tools like GitHub Actions use textual logs and 2D dashboards that can make it challenging to visualize temporal relationships and concurrency patterns across multiple pipeline runs.

This dissertation explores how Virtual Reality can improve CI/CD workflow visualization. Using a factory metaphor, pipeline data is translated into immersive 3D environments: workflows become floors, with their corresponding workflow runs represented as blocks placed on each floor, jobs become sub-blocks within those runs, and time unfolds as spatial depth. Users can walk through the pipeline history and understand execution patterns through natural movement.

A working VR prototype was implemented using Unity that transforms GitHub Actions data into an interactive 3D environment. The system includes temporal mapping algorithms, parallel and sequential layout organization for concurrent and dependent runs and jobs, time compression for inactive periods, and spatial audio for failure detection. Users can explore workflow history, identify problems, and understand pipeline behavior through spatial navigation.

A user study compared the VR system against GitHub Actions' web interface using tasks for temporal navigation, failure detection, and debugging. In VR, users completed tasks more quickly and achieved higher success rates compared to when using the GitHub Actions interface. The VR system received higher usability scores, with participants finding it easier to use and less complex.

This research provides practical guidelines for immersive CI/CD visualization. It demonstrates that spatial 3D layouts help with overview tasks and temporal navigation, while traditional interfaces work better for detailed analysis.

Keywords: immersive analytics, virtual reality, CI/CD visualization, software engineering, GitHub Actions, spatial visualization, temporal patterns, failure detection

ACM Classification:

- Human-centered computing → Human computer interaction (HCI)
→ Interaction paradigms → Virtual reality
- Human-centered computing → Visualization → Visualization application domains → Information visualization
- Software and its engineering → Software creation and management
→ Software development process management

AI Usage Disclosure

Artificial Intelligence (hereafter AI) tools have been utilized in this dissertation.

ChatGPT (GPT-4), **Grammarly** and **GitHub Copilot** have been used to assist with writing improvement, code development, and technical problem-solving. Specific uses include debugging Unity C# code, suggesting implementation approaches, and assisting with LaTeX formatting.

All AI-generated content was reviewed and validated for relevance, appropriateness, and accuracy before incorporation into the final manuscript to maintain scholarly integrity of this research.

The author takes full responsibility for the content of the document.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation on Immersive visualization of CI/CD pipelines contributes to achieving sustainable development through technological innovation that enhances software engineering processes and promotes more efficient development practices. By improving how developers understand and debug CI/CD workflows, this work supports the advancement of modern digital infrastructure and productive employment in the technology sector.

More specifically, this work contributes to the following Sustainable Development Goals:

SDG 8 Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SGD	Target	Contribution	Performance Indicators and Metrics
8	8.1	Demonstrated improvement in CI/CD task completion times, directly contributing to sustained economic growth through enhanced developer productivity in software engineering workflows.	Task completion time reduction; Developer productivity metrics; Software delivery cycle efficiency
	8.2	Developed innovative VR-based approaches for software engineering tools, aiming to achieve higher economic productivity through advanced spatial visualization techniques.	VR technology adoption rates; Innovation patents/publications; Advanced HCI implementation metrics
9	9.1	Enhanced reliability of digital infrastructure by providing superior CI/CD pipeline monitoring and failure detection capabilities through immersive visualization.	Pipeline failure detection accuracy; Infrastructure reliability scores; System improvements
	9.4	Reduced computational waste in development cycles by enabling faster identification of pipeline inefficiencies, promoting sustainable software engineering practices.	Computational resource savings; Failed build reduction; Energy efficiency in CI/CD processes
	9.5	Advanced scientific knowledge in immersive analytics through empirical validation of VR applications in software engineering, fostering technological capability development.	Published research contributions; Open-source tool adoption; Academic-industry collaboration metrics

Acknowledgements

I want to thank my parents and my sister for all the help they gave me, for believing in me, and for always being there for me throughout this journey. Your support and encouragement kept me going.

To my boyfriend, who is also my best friend, thank you for always being available to help me and for being the person who motivated me the most when I felt less motivated during my dissertation. Thank you for listening to me and for never letting me give up. I also want to thank your family for all their support and for making me feel welcomed.

To all my friends, thank you for the help and motivation. You made these years at FEUP so much better and are a big part of the reason I felt motivated to keep going. Thank you for all the good moments we shared.

Finally, I want to thank my supervisors, Rui Rodrigues and Daniel Mendes, because they were always available to help me from the beginning of the dissertation. Your patience, guidance and feedback made this work possible. Thank you for believing in my ideas and helping me make them real.

Anete Pereira

Contents

1	Introduction	1
1.1	Motivation	2
1.2	Problem Statement	2
1.3	Research Questions	3
1.4	Objectives	3
1.5	Document Structure	3
2	State Of The Art	5
2.1	Information Visualization	5
2.1.1	Two-Dimensional (2D) and Three-Dimensional (3D) Information Visualization	6
2.2	Immersive Analytics	8
2.2.1	Definition and Evolution	8
2.2.2	Key Concepts and Benefits	9
2.2.3	Sonification in Immersive Analytics	10
2.2.4	Toolkits and Systems	12
2.2.5	Challenges and Opportunities	13
2.3	CI/CD Pipeline Visualization	14
2.3.1	Continuous Integration and Continuous Delivery/Deployment	14
2.3.2	CI/CD Platforms and GitHub Actions	15
2.3.3	Immersive CI/CD Visualization Systems	16
2.4	Discussion	17
3	Immersive Simulator for CI/CD Pipelines	19
3.1	Problem Statement	19
3.2	CI/CD Execution Model	20
3.3	User Tasks and Visualization Requirements	22
3.4	Multimodal Design Considerations	23
3.5	Design Metaphor	24
3.6	Visual Encoding and Structural Mapping	26
3.7	Interaction Design Considerations	27
3.8	Summary	28
4	Prototype Implementation	29
4.1	System Architecture	29
4.2	Data Layer	31
4.3	Factory Visualization System	35
4.4	Audio Layer	46

4.5	Interaction Layer	49
4.6	Visual Encoding and Accessibility	52
4.7	Performance Considerations	56
4.8	Summary	57
5	Evaluation	59
5.1	Evaluation Setup	59
5.2	Participants	60
5.3	Task Design	62
5.4	Methodology	64
5.5	Results	68
5.5.1	Task Performance Analysis	68
5.5.2	System Usability Evaluation	71
5.5.3	Spatial Audio Assessment (Task 4)	72
5.5.4	Experience Correlation Analysis	74
5.5.5	Qualitative Observations	74
5.6	Discussion	77
6	Conclusions	81
6.1	Key Findings	81
6.2	Future Work	82
6.3	Concluding Remarks	84
	References	85

List of Figures

2.1	Examples of interactive visualizations created using D ³	6
2.2	Effect of visual design on time-series perception.	7
2.3	Stacking-based 3D visualization of radiation trajectories along a geographic route.	7
2.4	Examples of methods for linking 2D and 3D views.	8
2.5	Illustration of several recent Immersive Analytics systems.	10
2.6	A user interacts with an immersive 3D network visualization using a head-mounted display and controllers.	11
2.7	Visualization of urban noise propagation paths in VR.	12
2.8	Two examples of immersive visualization with IATK.	13
2.9	DXR enables rapid prototyping of immersive data visualizations.	14
2.10	The relationship between continuous integration, delivery, and deployment.	15
2.11	An overview of challenges and practices in adopting CI/CD and the relationship among them.	16
2.12	A VR-based pipeline visualization from VR-DevOps.	17
3.1	Hierarchy and dependency structure in GitHub Actions workflows, showing the relationship between the workflow definition (Listing 3.1) and its execution instances	22
3.2	Conceptual design metaphor showing the mapping of CI/CD elements to spatial representations in the building-inspired layout	25
4.2	Complete factory layout demonstrating the full visualization system implementation.	35
4.3	Timeline markers positioned at 30-minute intervals along the factory floor.	37
4.4	Three workflow floors showing vertical stacking with labels: "CI" (bottom), "Docusaurus" (middle), and "Update Cargo.lock" (top).	38
4.5	Workflow floor spatial arrangement from different perspectives.	39
4.6	Empty conveyor belt on a workflow floor.	40
4.7	A single workflow run visualized as a base block containing job blocks.	41
4.8	Workflow run status visualization through color coding.	42
4.9	Spatial encoding of workflow runs.	42
4.10	Job positioning within workflow runs.	44
4.11	Time compression portal labeling system.	45
4.12	Time compression portal spanning the factory environment.	46
4.13	Spatial audio visualization showing a sound icon above a failed workflow run.	47
4.14	VR controller input mapping for navigation	50
4.15	Hand-mounted user interface menu attached to the left hand.	51
4.16	Information panel system.	52
4.17	Comparison of text inspection with and without the magnifier tool.	52
4.18	Color accessibility validation results	54

4.19	Comparison of the outcome-based color schemes used for workflow runs and job elements.	55
4.20	Final color palette used in the prototype.	55
4.1	High-level architecture of the prototype system.	58
5.1	Evaluation setup showing participant interaction with the VR system.	61
5.2	Screenshots from the CI/CD tutorial video showing fundamental concepts and workflow structure.	65
5.3	Screenshots from the GitHub Actions tutorial video demonstrating interface navigation and status interpretation.	66
5.4	Screenshots from the VR system tutorial video showing interaction techniques and visual encoding principles.	66
5.5	Task completion time distributions across all evaluation tasks.	69
5.6	Task success rates by environment.	70
5.7	SUS score comparison between GitHub Actions and VR interfaces.	72
5.8	Task 4 success and failure distribution by user confidence levels.	74
5.9	Heatmaps showing mean SUS scores across different experience level combinations.	75
5.10	Heatmaps showing mean total task completion times across different experience level combinations.	76

List of Tables

3.1	Core CI/CD elements and key attributes.	20
3.2	Developer tasks and corresponding CI/CD elements.	23
3.3	Visual encoding mapping between CI/CD concepts and visual representations . .	27
4.1	Key technologies used in the implementation.	30
4.2	GitHub REST API endpoints used in the system.	32
4.3	Conveyor belt length calculation components	40
4.4	Spatial Audio Configuration for Failure Events	47
4.5	Failure Audio Source Lifecycle Events	48
4.6	Status-Based Color Encoding Used in the Prototype	54
5.1	Evaluation Session Structure	64
5.2	Data Collection Framework	67
5.3	Median (IQR) and Wilcoxon Signed Ranks test results for individual SUS ques- tionnaire items. The asterisk (*) indicates a significant difference between GitHub Actions and VR interfaces.	73
5.4	Median (IQR) responses to spatial audio evaluation questions on 5-point Likert scales.	73

Listings

3.1	Example GitHub Actions workflow showing the hierarchical structure	21
4.1	Unified data structure format for visualization	34

List of Acronyms

2D	Two-Dimensional
3D	Three-Dimensional
ACM	Association for Computing Machinery
API	Application Programming Interface
AR	Augmented Reality
CAVE	Cave Automatic Virtual Environment
CI	Continuous Integration
CD	Continuous Delivery/Deployment
DAG	Directed Acyclic Graph
DevOps	Development Operations
DXR	Data Explorer for XR
FIESTA	Free-roaming Immersive Environment to Support Team-based Analysis
HCI	Human computer interaction
HRTF	Head-Related Transfer Function
IA	Immersive Analytics
IATK	Immersive Analytics Toolkit
ID	Identifier
IQR	Interquartile Range
JSON	JavaScript Object Notation
MR	Mixed Reality
1 SD	Standard Deviation
SDGs	United Nations Sustainable Development Goals
SUS	System Usability Scale
UI	User Interface
URP	Universal Render Pipeline
VR	Virtual Reality
XR	Extended Reality
YAML	YAML Ain't Markup Language

Chapter 1

Introduction

Immersive technologies such as Virtual Reality (VR) have expanded far beyond entertainment, finding applications in data visualization, education, and software development. Immersive analytics, a subfield within this broader trend, investigates how VR and related interfaces enhance data analysis by leveraging spatial, interactive, and multisensory environments. Foundational research has shown that higher levels of immersion can enhance users' spatial understanding and performance in complex tasks [2], while more recent work in immersive analytics has applied these findings to data exploration and decision-making [32]. Additional studies have shown that VR can help users build a more intuitive and embodied understanding of data by engaging their senses and spatial awareness in ways that go beyond traditional charts and graphs [25].

A field in which a large volume of data is being generated is the Software Engineering industry, where the continuous building, testing, and deployment of software produces substantial amounts of process-related information. In particular, the adoption of Continuous Integration and Continuous Deployment (CI/CD) practices has become common in modern development workflows, contributing to the substantial amounts of process-related information produced. These pipelines automate the building, testing, and deployment of code changes to improve software quality and delivery speed [39]. GitHub Actions, a cloud-native CI/CD service, allows developers to define workflows that respond to repository events, such as pushes or pull requests. While the adoption of GitHub Actions is growing, developers often face challenges related to YAML configuration complexity and workflow maintenance [37]. Despite these advances, existing tools for inspecting and debugging GitHub Actions workflows rely heavily on 2D dashboards and textual logs. Understanding temporal patterns, repeated failures, or performance regressions becomes cognitively demanding as workflows grow in complexity and historical depth. This research aims to enhance the accessibility and intuitiveness of CI/CD pipeline visualization, especially for developers new to these systems, by utilizing the spatial and interactive capabilities of immersive environments.

1.1 Motivation

Given the learning curve associated with CI/CD tools and the complexity of their traditional interfaces, this research deliberately focuses on improving the experience for developers starting to work with these systems. Rather than targeting professional programmers who have already developed expertise with existing tools, the investigation prioritizes making CI/CD concepts more approachable and intuitive for newcomers to the field. Conventional CI/CD tools like GitHub Actions are helpful in inspecting individual workflow runs but offer limited support for exploring broader patterns across time, branches, or contributors. For example, comparing workflows between feature branches or identifying recurring failures often requires developers to switch between logs, dashboards, and tables, forcing them to reconstruct relationships and trends manually. This dissertation is motivated by the potential for VR environments to make these connections more intuitive. Spatial layout, color, interaction, and sound can be used to represent job status, duration, and structure in a way that supports:

- Visual comparison of workflows over time or across branches.
- Faster recognition of failed jobs or long-running builds.
- Pattern detection through grouping, scale, and proximity.

The availability of real GitHub Actions data in structured JSON format enables the creation and testing of immersive systems using real-world complexity without needing artificial or simplified datasets.

1.2 Problem Statement

CI/CD workflows like those supported by GitHub Actions are powerful tools for automating software integration and deployment. However, their growing complexity poses challenges regarding visibility, traceability, and historical understanding. Studies have shown that one of the key difficulties in adopting continuous practices is maintaining team awareness and transparency, especially when diagnosing build and test results across time and multiple branches [39]. While GitHub Actions provides logs and dashboards to inspect individual runs, it lacks dedicated tools for temporal pattern recognition, failure propagation analysis, or comparative inspection across workflows. These limitations make it harder for developers to detect recurring issues, identify regressions, or understand how workflows evolve, particularly in large-scale or long-lived projects [39]. While immersive visualization techniques have demonstrated benefits for understanding complex structured data, their application to CI/CD pipelines remains limited. Current GitHub Actions tools are primarily focused on 2D textual views, providing minimal support for interactive, overview-driven analysis that could enhance contextual understanding of workflow behavior over time.

1.3 Research Questions

This dissertation investigates how immersive VR environments can support understanding CI/CD workflows through structured visualization. The following research questions guide the development and evaluation of the system:

- **RQ1:** Does a VR-based visualization system help users more easily identify and interpret CI/CD workflow structure and failures compared to a traditional 2D interface?
- **RQ2:** Does using spatial layout in a 3D environment contribute to reduced task time and fewer errors when exploring workflow execution data?
- **RQ3:** How do users perceive the ease of use, learnability, and overall satisfaction with the VR-based visualization system compared to traditional tools?

1.4 Objectives

This dissertation explores how immersive virtual reality can support the visualization and interpretation of CI/CD pipelines, focusing on workflows executed through GitHub Actions. The main goals are:

- Design and implement a VR-based prototype for visualizing GitHub Actions workflows using real structured data;
- Create a spatial 3D layout that enables users to explore workflow runs and job structures interactively;
- Include auditory cues as an additional layer of feedback, particularly for failed jobs;
- Conduct a user study comparing the VR system to the standard GitHub Actions interface, assessing usability and perceived effectiveness across common CI/CD debugging tasks.

1.5 Document Structure

This dissertation is organized into five chapters, each addressing a key aspect of the research.

Chapter 1 introduces the background, motivation, problem statement, objectives, and research questions that guide this work.

Chapter 2 reviews the existing literature and related work in immersive analytics, CI/CD visualization, and Virtual Reality, setting the context for the proposed solution.

Chapter 3 describes the conceptual design.

Chapter 4 details the technical details, and implementation of the Immersive Simulator for CI/CD Pipelines.

Chapter 5 details the user study methodology, tasks, collected data, and analysis performed to assess the effectiveness and usability of the prototype.

Finally, Chapter 6 summarizes the main findings, discusses limitations, and suggests directions for future research.

Chapter 2

State Of The Art

This chapter presents a comprehensive review of the research domains that inform the development of immersive CI/CD pipeline visualization systems. The review is structured to build understanding from foundational concepts to specific applications, ultimately identifying gaps that motivate the research contributions of this thesis.

2.1 Information Visualization

Information visualization is the use of visual representations of abstract data to support human cognition and decision-making. Its primary goal is to transform data into graphical formats, allowing the human visual system to detect patterns, trends, and outliers [29]. The concept of “amplifying cognition” through visual means was introduced by Card, Mackinlay, and Shneiderman in their foundational book *Readings in Information Visualization: Using Vision to Think* [10], where they describe how visual artifacts can support memory, inference, and perceptual processing. This idea has been reiterated and expanded upon in later literature, including survey work on recent advances in interactive visualization systems [29].

This field is particularly valuable in large-scale, high-dimensional, or dynamic data where raw numeric or textual formats are insufficient to reveal structure. With the rise of big data and real-time analytics, information visualization has become crucial across domains like business intelligence, scientific research, and education. In these areas, visualization helps users make sense of large, dynamic, or high-dimensional datasets that would be difficult to interpret using raw tables or text [29].

Research has consistently shown that well-designed visualizations enable faster and more accurate understanding compared to non-visual formats [29, 19]. Modern web-based tools such as D³.js, Plotly, Vega-Lite and many others exemplify the progress made in interactive information visualization, allowing developers and analysts to build responsive and dynamic visual interfaces [6, 38, 42]. Figure 2.1 shows examples of visualizations generated with D³, illustrating the diversity and flexibility of these approaches.

Furthermore, recent research has explored hybrid visualization strategies that combine two-dimensional and three-dimensional elements, each supporting distinct cognitive tasks. As the field of information visualization continues to evolve, new strategies are emerging to accommodate the growing complexity of data and user tasks. One such direction involves integrating 2D and 3D elements to support spatial reasoning, structural clarity, and improved navigation, topics explored in more detail in the following section.

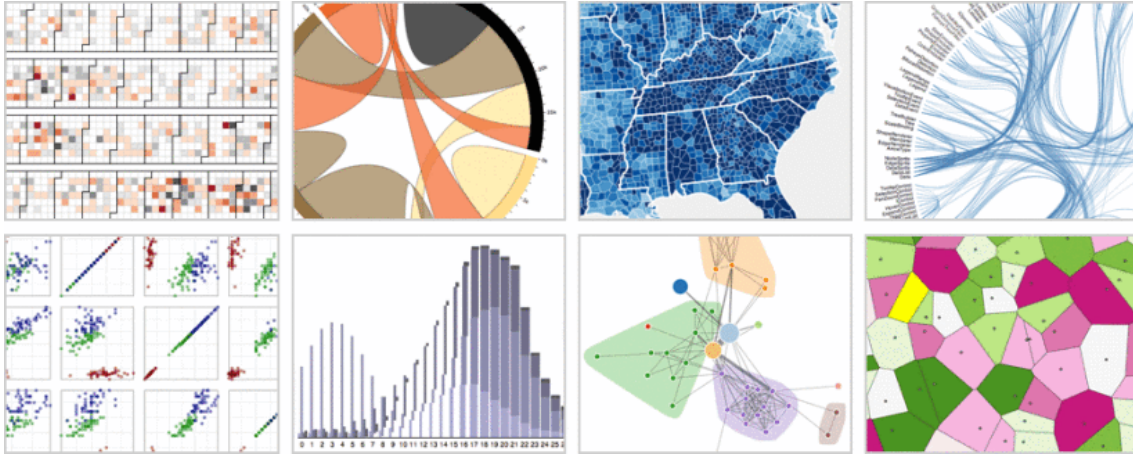


Figure 2.1: Examples of interactive visualizations created using D3 (source: [6]).

2.1.1 Two-Dimensional (2D) and Three-Dimensional (3D) Information Visualization

Two-dimensional (2D) visualizations have long been the foundation of data analysis. Examples include bar charts, line graphs, scatter plots, heat maps, and treemaps. These techniques are popular because they are straightforward, easy to read, and excellent at supporting tasks like comparison, trend detection, and categorization [29]. Figure 2.2 illustrates how small changes in noise levels and layout can influence the readability of 2D plots such as scatter plots and line graphs.

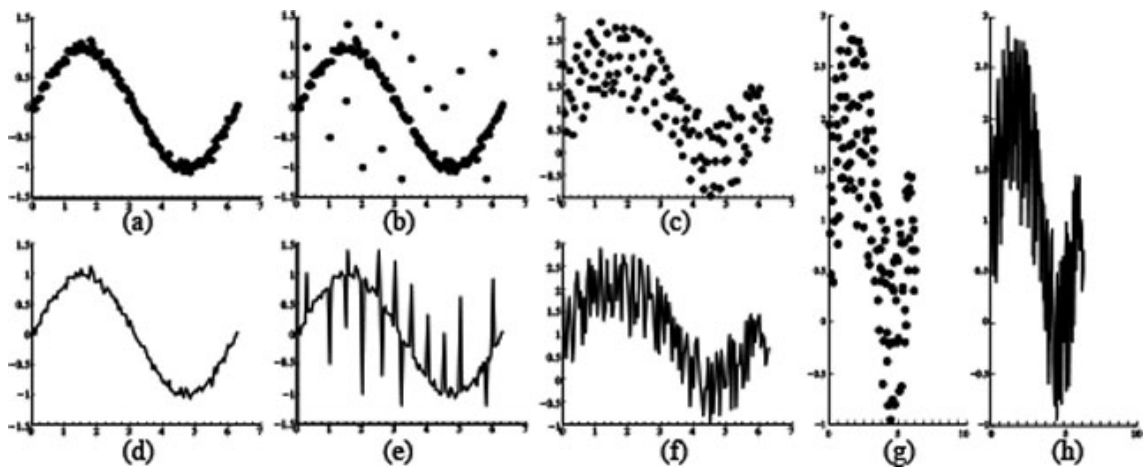


Figure 2.2: Effect of visual design on time-series perception: scatter plots (a–c, g) and line graphs (d–f, h) perform differently depending on data noise, outliers, and display aspect ratio. Source: [45].

Despite their strengths, 2D techniques face limitations when applied to high-dimensional, hierarchical, or spatially complex data. Reducing data to two dimensions often leads to loss of structure or distortion [43]. Three-dimensional (3D) visualization introduces depth as an extra visual cue. This visualization has been used to represent complex structures, like data mountains, hierarchies (e.g., cone trees [35]), and network graphs in ways that can improve structural understanding and spatial reasoning [43]. These representations are particularly helpful in tasks requiring overviews of complex relationships or navigation through dense information spaces. An example of a 3D representation is shown in Figure 2.3 where two of the dimensions are used to represent the geographic map and the third to stack trajectories and detailed attribute data.

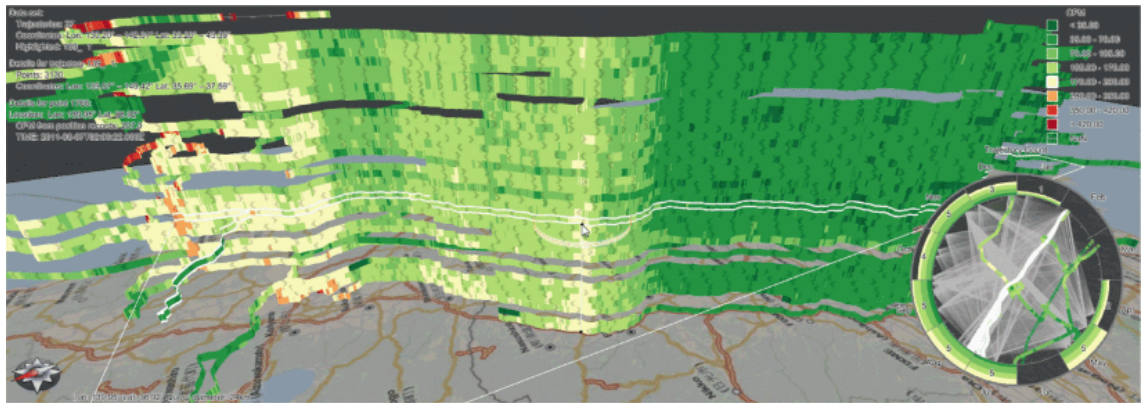


Figure 2.3: Stacking-based 3D visualization of radiation trajectories along a geographic route. Two spatial dimensions encode location, and height encodes attribute value (radiation): the interface shows radiation values along the Tokio-Fukushima highway. (source: [44]).

However, 3D visualization comes with trade-offs. Challenges such as occlusion, perspective distortion, and the need for spatial navigation can increase cognitive effort and reduce precision in fine-grained comparisons [20, 19]. As a result, its effectiveness is highly task-dependent: while

3D visualizations can enhance spatial understanding and exploratory analysis, studies have shown that 2D representations often deliver better or equivalent results for tasks involving precise judgment or comparison [14]. To address the limitations of both approaches, hybrid visualizations combining 2D and 3D elements have been proposed. These strategies aim to balance expressive power and usability, enabling users to switch between or coordinate perspectives as needed. A recent survey highlights design patterns for combining 2D and 3D views through spatial linking, interaction control, and animated transitions [19]. These approaches are particularly valuable in domains involving multiscale, temporal, or structured data. These hybrid strategies are illustrated in Figure 2.4, which shows different linking methods between 2D and 3D views; visual alignment, coordinated interaction, and animated transitions.

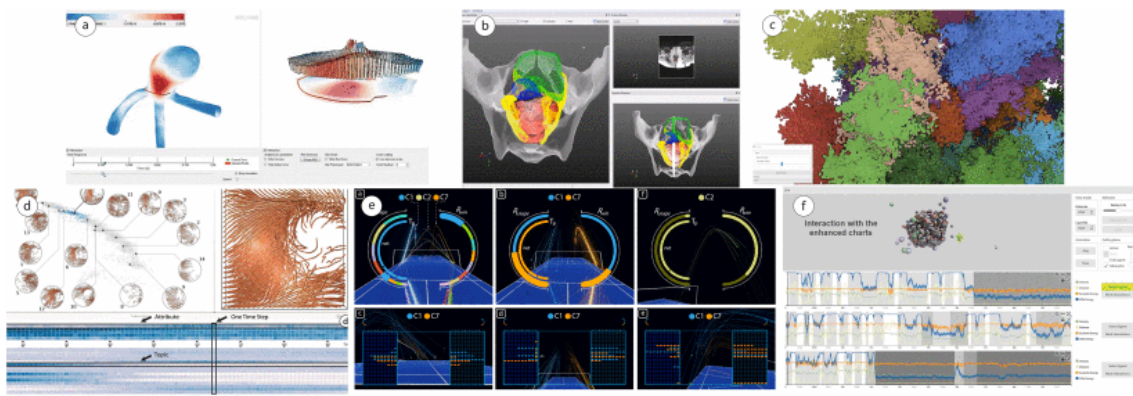


Figure 2.4: Examples of methods for linking 2D and 3D views: visually (top row), interactively (middle row), and through animation (bottom row). Source: [43].

In summary, neither 2D nor 3D visualization is inherently superior. Instead, the suitability of each approach depends on the data characteristics and the analytical goals of the user. Hybrid methods continue to gain traction as a flexible compromise in modern visualization systems.

2.2 Immersive Analytics

Immersive analytics (IA) represents a significant evolution in data visualization, extending beyond traditional 2D and 3D techniques by embedding users within data environments that support full-body interaction, multisensory feedback, and a strong sense of presence through technologies like Virtual Reality (VR), Augmented Reality (AR), and Mixed Reality (MR). These settings allow users to interact with data as if it were part of the physical world, offering new pathways for exploring complex, multiscale, and dynamic datasets.

2.2.1 Definition and Evolution

Immersive Analytics (IA) is the use of immersive technologies, such as VR, AR, and MR, to enhance data exploration, interpretation, and decision-making. It aims to support analytical reasoning through embodied interaction, spatial cognition, and multisensory feedback [15, 34]. Im-

mersive visualization expands the scope of traditional and 3D visualization by introducing these elements and placing the user inside the data space [24]. These immersive environments allow users to navigate and manipulate data, unlocking new modes of understanding for complex, multiscale, and dynamic datasets [24]. The foundations of immersive analytics emerged in the early 2010s but were strongly influenced by much earlier work in scientific visualization and virtual reality from the 1990s. For instance, Bryson's work on data spaces in VR [9], Cruz-Neira et al.'s CAVE system [12], and Bowman et al.'s early research on 3D interaction for scientific data [7] are often cited as conceptual and technological forerunners of IA. These early examples highlighted spatial metaphors and depth cues to support cognitive performance. Over time, improvements in head-mounted displays, motion tracking, and spatial interaction frameworks allowed immersive analytics to take shape as a distinct research area. Fonnet and Prié offered a historical overview of this transition and identified how IA sits at the intersection of InfoVis and immersive systems [16, 24]. Immersive analytics is not simply about visual fidelity; it reconfigures the relationship between user and data. While traditional visualizations maintain a separation between observer and display, immersive systems enable users to inhabit the visual space. This embodiment enhances spatial cognition, attention, and memory [23] and supports intuitive exploration using motion, gestures, and head orientation [24].

The user-centered framework emphasizes this shift: immersive analytics is not only about fidelity but about interaction, perspective, and situated cognition, treating users as participants rather than external viewers [34]. This diversity of interaction styles is reflected in several immersive systems developed in recent years, combining VR, AR, and tangible interfaces for different domains and user contexts (see Figure 2.5)

2.2.2 Key Concepts and Benefits

Immersive analytics environments distinguish themselves through several interrelated concepts that differentiate them from conventional approaches:

Spatialization: Information is mapped into 3D space, enabling users to perceive data structures and relationships through proximity, orientation, and motion. Spatial layouts can improve memory retention and provide intuitive navigation through data [23]. A typical immersive setup includes handheld controllers, head tracking, and spatial input modalities for manipulating 3D data layouts, as seen in Figure 2.6.

Embodiment and Presence: Users feel physically present in the environment, manipulating visual elements through gestures and body movement. This sense of "being there" can enhance task engagement, especially in exploratory or multiscale contexts [34].

Multimodal Interaction: Beyond vision, immersive systems frequently integrate sound (e.g., sonification), haptics, or other feedback channels. Sonification and spatial audio, for instance, have been shown to direct attention, encode time-based changes, or guide users in large datasets [4, 8].

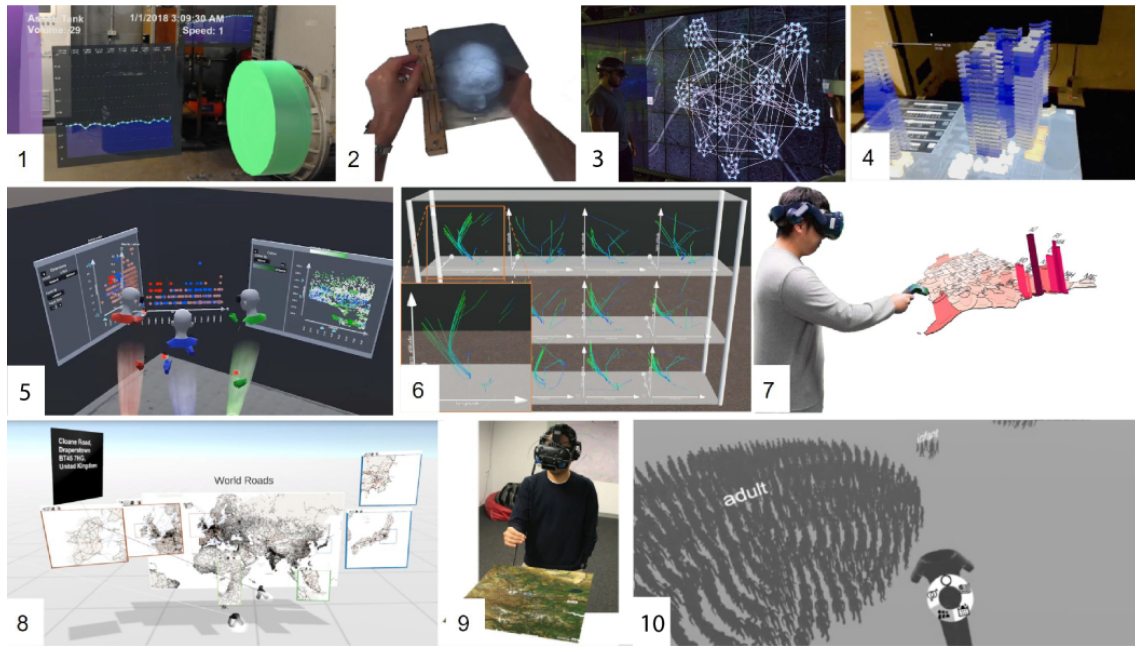


Figure 2.5: Illustration of several recent Immersive Analytics systems: (1) Corsican Twin, (2) the Embodied Axes controller for precise interaction with 3D visualizations in AR, (3) AR personal navigation of networks on wall-size displays, (4) the UpLift system for building energy analytics, (5) the FIESTA system for collaborative immersive analytics in VR [26], (6) a study of small-multiples in VR, (7) the Tilt Map system for 3D Choropleth maps visualization, (8) a multiscale navigation system in AR, (9) bare hand interactions in AR maps, and (10) immersive unit visualization [15].

Real-World Metaphors: Immersive systems may adopt spatial metaphors grounded in everyday experience, such as architectural layouts or mechanical structures, to increase intuitiveness. This strategy is commonly used in abstract data visualization to foster sensemaking [24].

These techniques have shown benefits across several domains. For example, immersive network visualizations allow users to understand hierarchy and connectivity more effectively than in 2D [21, 23]. Similarly, IA has also been applied to urban data visualization [4] and biological data like cell tracking

2.2.3 Sonification in Immersive Analytics

Sonification is the use of non-speech audio to convey information or highlight patterns in data [5]. In immersive analytics, it complements visual representations by offering an additional sensory channel, well suited for directing attention, encoding temporal dynamics, and reducing visual overload. Sonification becomes especially powerful when paired with spatial audio in immersive environments, allowing users to locate and interpret events using directional cues and sound properties like pitch or rhythm [8, 4].

Research suggests that sonification can improve engagement and performance in data-rich environments, especially where continuous monitoring or multitasking is required [4, 8]. For



Figure 2.6: A user interacts with an immersive 3D network visualization using a head-mounted display and controllers. A virtual laser pointer is implemented to highlight and select nodes. Users can scale and relocate the network using pinch-to-zoom interaction with both controllers. The figure shows the karate network in a 3D layout [23].

example, in immersive robot teleoperation, spatial audio cues have been shown to reduce task errors and guide user attention to off-screen events [8]. In urban analytics, sonification has helped users explore the diffusion of noise pollution by layering sound on top of visual representations of acoustic phenomena [4].

Sonification also enhances temporal perception in immersive systems. When visualizations encode complex or asynchronous workflows, sound can help communicate the timing, duration, and sequence of events. For instance, rising pitch or rhythmic pulses can indicate acceleration or synchronization across stages [17]. These auditory cues reduce dependence on spatial awareness alone and offer redundancy to support users with visual fatigue or cognitive overload.

Despite these advantages, most current immersive systems use sonification in limited or *ad hoc* ways—often just as alarms or ambient sounds. Systematic design approaches and user-driven mapping strategies remain an open area for research [4]. As immersive analytics matures, integrating sound as a first-class information modality, rather than a supplementary one, may prove essential for making sense of dynamic, multiscale, or time-sensitive data.

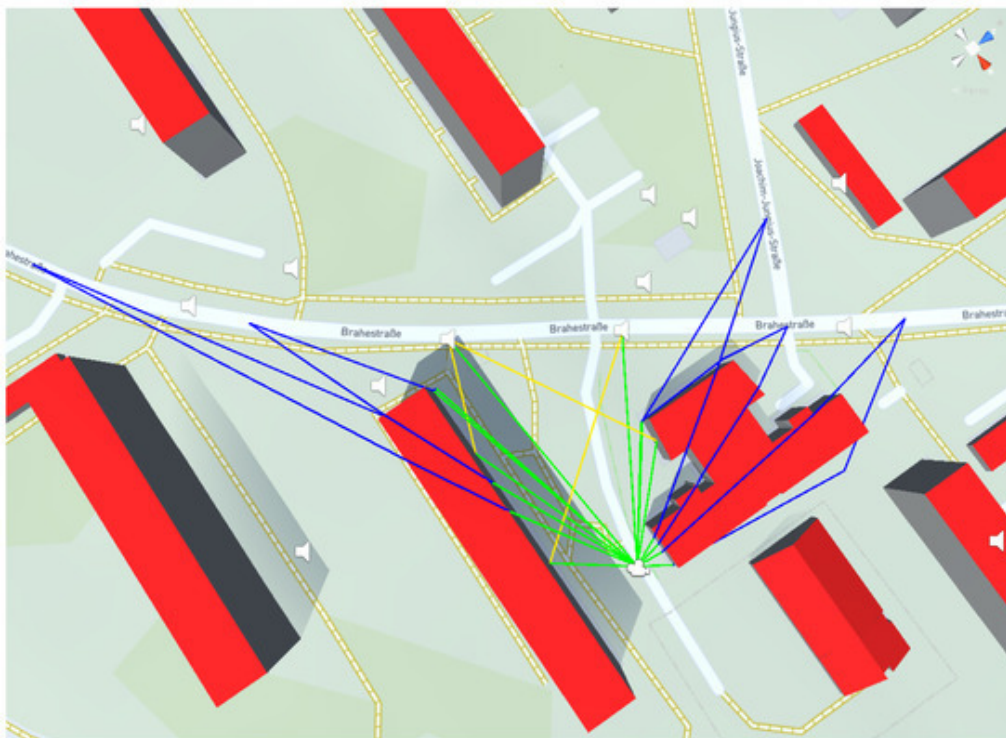


Figure 2.7: Visualization of urban noise propagation paths in VR. Green lines show direct sound paths, blue indicates diffraction, and yellow shows reflections. This multimodal approach integrates visual and auditory feedback for spatial reasoning about acoustic behavior. Source: [4]

2.2.4 Toolkits and Systems

A limited but growing number of toolkits support the development of immersive analytics systems by abstracting low-level rendering and interaction components. Fonnet and Prié (2021) summarize several of these efforts and emphasize the importance of reusable frameworks in accelerating immersive analytics research [16]. An example is the Immersive Analytics Toolkit (IATK) [11], which provides Unity components for creating immersive visualizations with abstract datasets. IATK has been used in academic research to implement spatial data exploration and 3D timelines [24], though its adoption remains mostly within prototyping contexts. Two use cases of IATK are shown in Figure 2.8, demonstrating interactive brushing and linked views in immersive and augmented environments.

Another notable system is DXR (Data Explorer for XR), which is designed for immersive scientific visualization and emphasizes flexible integration with high-dimensional data sources [41]. DXR provides an architecture for building VR visualizations, including timeline-based and spatiotemporal layouts. Figure 2.9 shows various immersive visualizations created with DXR, including 3D bar charts and spatial narratives.

However, no single dominant toolkit exists. Many systems are still built directly on game engines (Unity, Unreal) or web-based XR frameworks customized for specific domains.

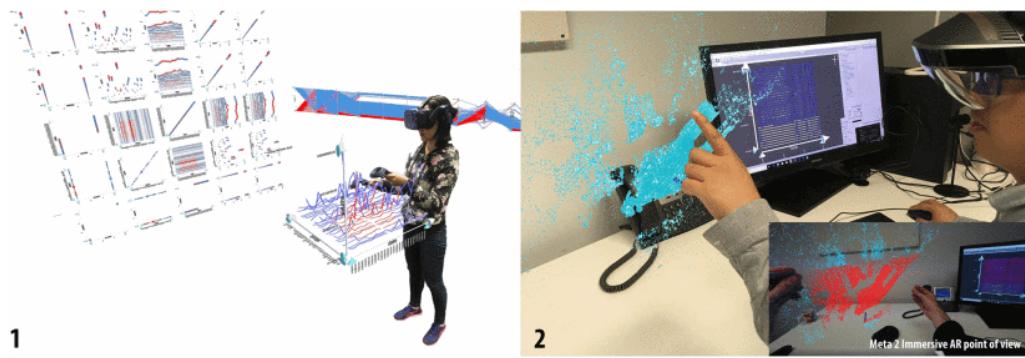


Figure 2.8: Two examples of immersive visualization with IATK: (1) immersive VR visualization of multidimensional financial data using brushing to link multiple views; (2) an augmented desktop setup with a 3D map above a physical workspace linked to 2D displays [11].

2.2.5 Challenges and Opportunities

While immersive analytics systems offer rich and engaging ways to explore data, they also present several technical and cognitive challenges. One common issue is user fatigue caused by prolonged use of head-mounted displays and spatial interfaces. Motion sickness, limited display resolution, and occlusion of visual elements can further reduce effectiveness in long sessions [15]. Additionally, immersive environments may impose a higher cognitive load due to the need to spatially parse large datasets, especially in 3D layouts with overlapping elements [34].

Another barrier is the lack of standardization in tools and evaluation methods. Most current systems are customized or built on game engines with limited reusability, restraining broader adoption. A review by Choe et al. highlights that immersive analytics often lacks standardized user testing protocols, relying instead on small-scale studies with *ad hoc* tasks and subjective feedback [18]. This limits comparability and generalization across systems.

Despite these challenges, several opportunities exist. Improvements in standalone VR/AR hardware, hand tracking, and wireless systems reduce friction in setup and mobility, expanding the range of possible deployment scenarios from collaborative analytics to on-site industrial use, among others. [15].

A typical challenge in immersive data visualization is conveying temporal structure and sequential logic in datasets that unfold over time. Many existing systems focus on the spatial representation of data attributes but offer limited support for visualizing temporal flows or asynchronous parallelism [23]. This becomes particularly relevant when users need to understand both global execution patterns and fine-grained task dependencies. Designing spatial-temporal mappings that balance clarity, scale, and narrative coherence remains an open research opportunity. Addressing some of these challenges, especially in conveying temporal flow and execution structure in complex technical domains, remains a key area of investigation and motivates the system introduced in the following chapter.

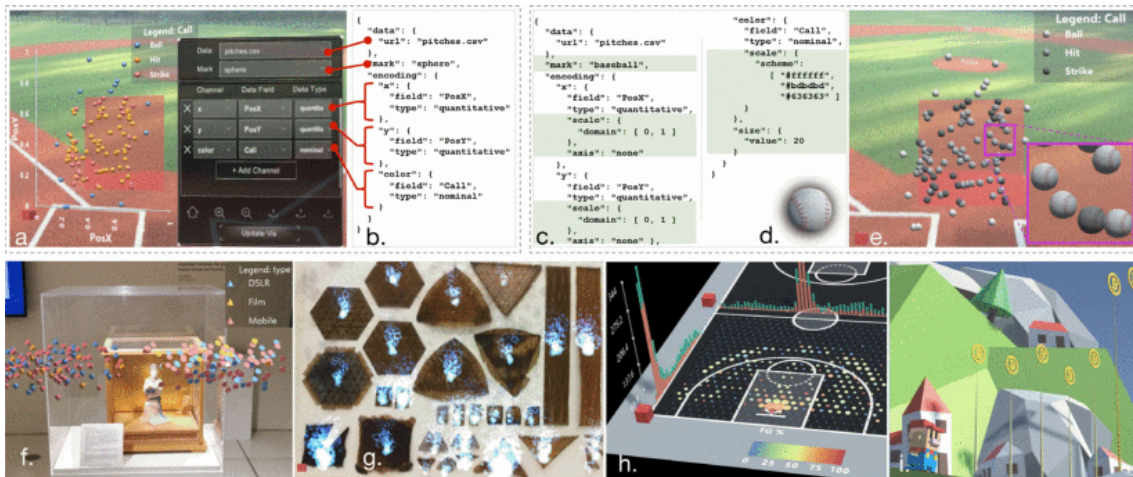


Figure 2.9: DXR enables rapid prototyping of immersive data visualizations: (b, c) declarative specifications represent visualizations; (a: right) DXR’s GUI inside VR enables parameter control; (d-e) custom 3D graphical marks; (f-i) examples including vector fields, decaying materials, basketball analytics, and 3D currency timelines [41].

2.3 CI/CD Pipeline Visualization

Continuous Integration (CI), Continuous Delivery, and Continuous Deployment (collectively referred to as CI/CD) are modern software engineering practices that aim to automate code integration, testing, and deployment in order to reduce errors and accelerate delivery cycles [39]. A CI/CD pipeline manages a series of tasks, from code commits to deployment, offering quick feedback to developers and encouraging continuous improvement. However, as pipelines grow more complex, visualizing and comprehending their execution becomes challenging. This section explores the CI/CD visualization field, highlighting standard practices and challenges across various platforms. It examines the common visualization approaches used by major CI/CD systems and identifies gaps in current interfaces. The section then focuses on GitHub Actions as a representative platform for detailed analysis, given its widespread adoption and typical visualization challenges. Finally, it reviews initial attempts at immersive CI/CD visualization that have emerged in recent research.

2.3.1 Continuous Integration and Continuous Delivery/Deployment

CI/CD are interrelated practices that automate the process of building, testing, and releasing software changes. CI focuses on integrating code into a shared repository frequently, often multiple times a day, with automated builds and tests validating each change. CD extends this process by automating the preparation of these builds for release, while Continuous Deployment goes further by automatically deploying validated builds into production environments without human intervention [39]. Figure 2.10 illustrates the conceptual relationship between CI, CD, and Continuous Deployment, as presented in the original study. It highlights how these practices are sequenced and interdependent.

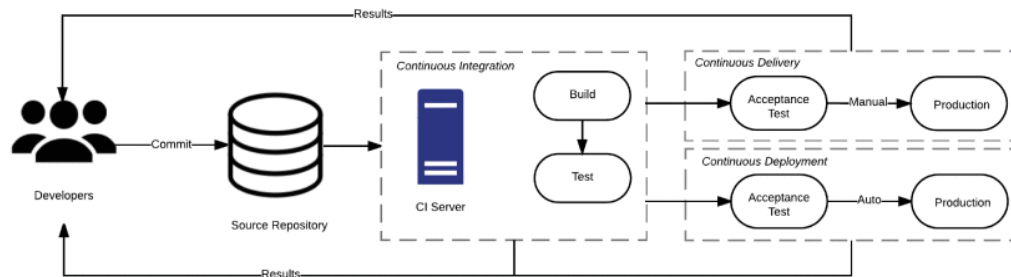


Figure 2.10: The relationship between continuous integration, delivery, and deployment. Source: [39]

These practices aim to increase software quality, shorten feedback loops, and reduce the cost of delivering changes. However, their adoption introduces architectural and organizational complexity. Figure 2.11 shows a synthesis of challenges and practices in CI/CD adoption, mapping interdependencies between problems and mitigation strategies. This visualization provides a broader view of developers' structural complexity in real-world deployments.

CI/CD pipelines often consist of multiple interdependent jobs and stages, with tasks executing sequentially and in parallel. As pipelines evolve, they become increasingly dynamic and challenging to monitor, especially in projects with high release frequency or multiple contributors. Research highlights that developers must often interpret the state of builds and deployments through a mix of log files, console outputs, and dashboard views, which are typically fragmented and text-heavy [1]. Several studies have emphasized the role of visualization in addressing these issues. Visual representations of pipeline states, execution flows, and failure points can help reduce cognitive load and improve awareness of system behavior [1, 13]. Despite this, many current tools fail to support these needs in large-scale or rapidly evolving environments adequately.

2.3.2 CI/CD Platforms and GitHub Actions

The CI/CD landscape encompasses several major platforms, each with distinct characteristics and adoption patterns. **Jenkins**, one of the earliest and most established platforms, offers extensive plugin ecosystems and self-hosted flexibility but requires significant configuration overhead. **GitLab CI/CD** provides integrated DevOps capabilities within GitLab's unified platform, supporting both cloud and self-managed deployments. **Azure DevOps** delivers comprehensive Microsoft ecosystem integration with Azure Pipelines for CI/CD automation. **CircleCI** and **Travis CI** focus on cloud-native approaches with streamlined configuration for GitHub repositories. **Bitbucket Pipelines** offers Atlassian ecosystem integration, while newer platforms like **GitHub Actions** have rapidly gained adoption.

GitHub Actions, introduced in 2018 as part of the GitHub ecosystem, has emerged as a particularly significant platform due to its integration with repositories, support for declarative YAML configurations, and access to a marketplace of reusable actions [36]. GitHub Actions allows developers to define and automate workflows that span testing, deployment, notifications, and other tasks, making it a flexible and powerful tool for modern software pipelines.

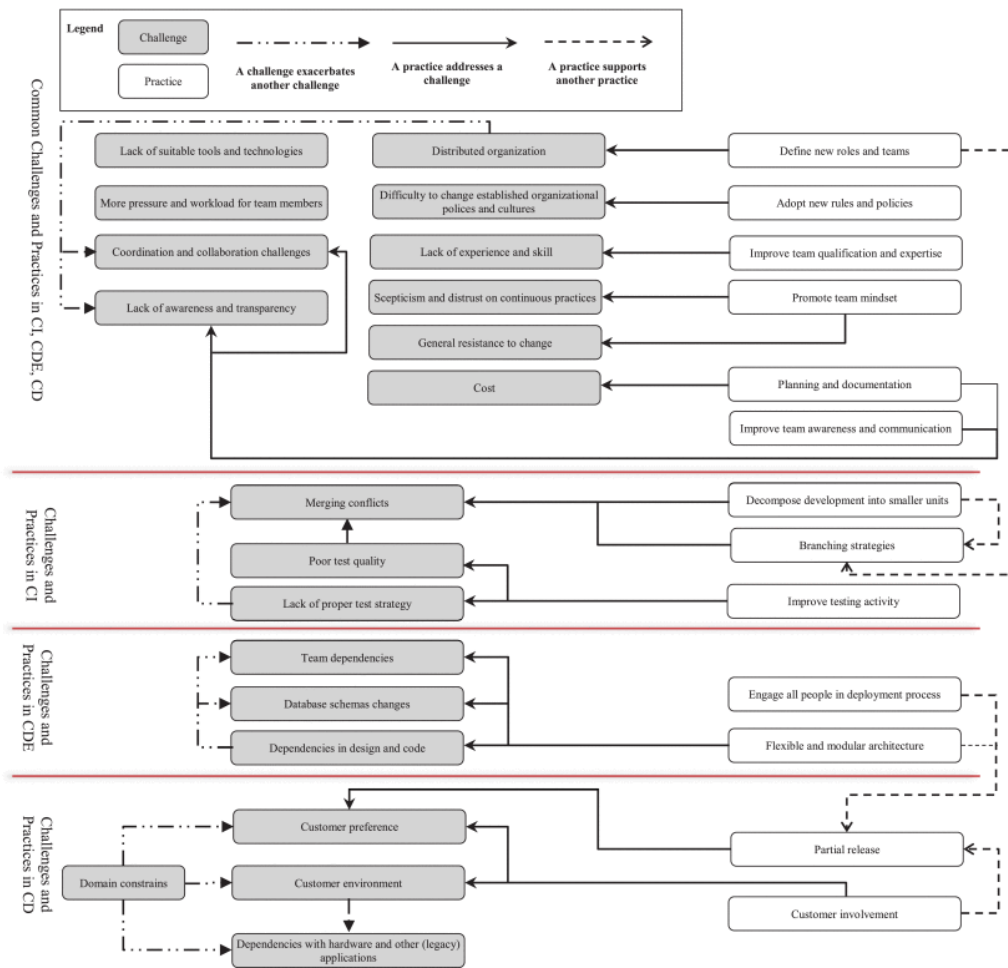


Figure 2.11: An overview of challenges and practices in adopting CI/CD and the relationship among them. Source: [39]

Despite its widespread use, developers face difficulties understanding and debugging workflow executions, especially in large or highly parallelized pipelines. Challenges include interpreting lengthy log files, lacking a clear temporal overview, and the absence of visual representations of job dependencies and statuses [36, 22]. These issues mirror those in other CI/CD systems, where visual and structural gaps impede situational awareness and effective troubleshooting [46]. Among the various CI/CD platforms, GitHub Actions has gained particular traction due to its seamless integration with the world's largest code hosting platform.

2.3.3 Immersive CI/CD Visualization Systems

To address the limitations of traditional CI/CD interfaces' limitations, a few exploratory systems have applied immersive and spatial visualization techniques to this domain. These efforts build upon research in immersive analytics, which shows that spatial metaphors and embodied interaction can enhance the comprehension of complex workflows [15, 23].

One example is VR-DevOps, a virtual reality prototype designed to visualize DevOps pipelines in immersive 3D space. The system organizes pipeline stages using spatial layout and supports interaction for navigation and anomaly identification [33]. Figure 2.12 shows a visualization of the VR-DevOps system with pipeline runs as vertical planes colored by status, stages and internal steps are displayed in a structured manner that facilitates spatial reasoning and visual comparison across executions [33].

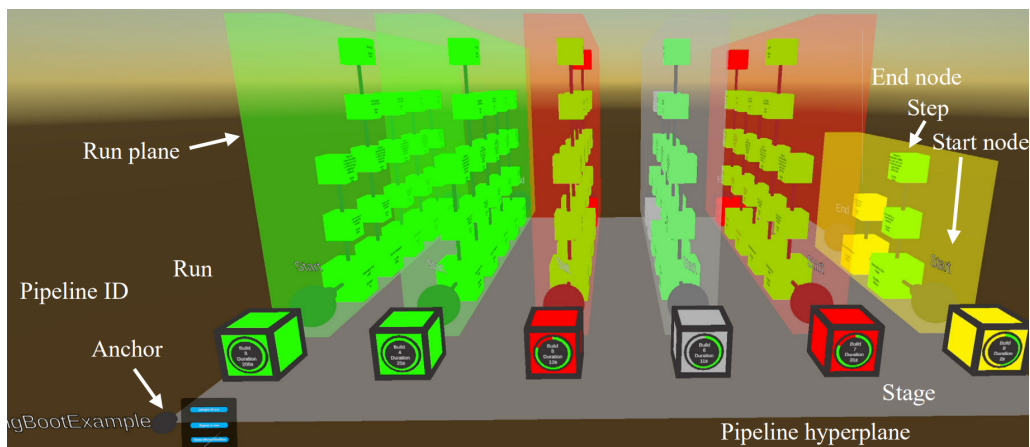


Figure 2.12: A VR-based pipeline visualization from VR-DevOps, showing colored run planes, execution stages, and internal steps. Source:[33]

While VR-DevOps illustrates the feasibility of immersive pipeline exploration, its implementation targets relatively small-scale workflows and has been evaluated only in preliminary usability studies. Current immersive systems often do not address the full complexity of CI/CD workflows. Open challenges include representing parallelism and asynchrony, encoding temporal sequences, and supporting real-time feedback. These limitations reflect known research problems in immersive visualization more generally, particularly concerning scalability and semantic clarity [15, 23].

2.4 Discussion

The literature on information visualization, immersive analytics, and CI/CD systems reveals gaps in how complex, temporal workflows are represented and explored. Traditional CI/CD tools rely heavily on textual logs and fragmented dashboards, which slow developers' ability to understand execution flow and locate failures [1, 36, 22]. Many visualization tools aim to clarify pipeline states but fail to support forming a cohesive mental model, particularly in dynamic or parallel execution contexts.

Immersive analytics introduces spatial, embodied, and multimodal strategies that have shown benefits in other domains, such as scientific visualization and robotics [15, 23, 8]. Applied to CI/CD, these techniques offer the potential for more intuitive representations of hierarchy, temporal structure, and concurrency. However, most existing immersive systems focus on abstract or

domain-specific datasets and do not address software engineering workflows or temporal data at scale.

Key challenges include managing asynchronous tasks, encoding temporal logic, and maintaining readability in parallelism. Developers struggle with overloaded log outputs and lack visual support for tracing dependencies across stages [36]. These gaps suggest the need for systems that integrate:

- Spatial hierarchy to separate workflows and jobs,
- Temporal mapping for sequence and duration,
- Consistent status encoding (e.g., success/failure),
- Interaction techniques for navigation and filtering,
- Compression strategies to manage scale,
- Multimodal feedback to guide attention and reduce overload.

Sonification remains underused despite evidence that spatial audio can support event localization and temporal perception in immersive environments [4]. Most current systems use sound only for alerts, missing opportunities to integrate it into the core analytic process.

Overall, the intersection of immersive analytics and CI/CD remains underexplored. There is a clear opportunity to develop systems that map software pipelines' temporal and structural features into immersive environments, improving navigation, comprehension, and debugging. The following chapter presents a proposed approach to address these challenges through spatial and temporal structuring, interaction mechanisms, and multimodal feedback.

Chapter 3

Immersive Simulator for CI/CD Pipelines

The previous chapter identified key limitations in existing CI/CD visualization tools, particularly their inability to represent temporal progression, job concurrency, and hierarchical relationships in an integrated way. Although immersive analytics has shown promise in other domains, its use in CI/CD contexts remains underexplored.

This chapter presents a conceptual framework for visualizing GitHub Actions workflows in an immersive environment. It begins with a problem statement outlining developers' core challenges when interpreting complex CI/CD pipelines (Section 3.1). Then, it defines the key execution elements of GitHub Actions and their structural and temporal semantics (Section 3.2). After this, it identifies common developer tasks and derives visualization requirements to support these activities (Section 3.3). It then discusses multimodal design considerations for combining visual and auditory feedback (Section 3.4). Next, it introduces a spatial metaphor that maps these elements into a 3D representation (Section 3.5). The chapter then outlines the visual encoding principles for communicating execution structure, status, and timing (Section 3.6). It concludes with interaction goals to support practical analysis and navigation within the system (Section 3.7).

3.1 Problem Statement

CI/CD pipelines, such as those implemented with GitHub Actions, play a major role in automating software development workflows. However, as these pipelines grow in complexity and concurrency, developers face increasing difficulty understanding their structure and behavior. Challenges include diagnosing failures across distributed jobs, tracking the evolution of workflows over time, and comparing execution outcomes across branches or commits.

Existing tools for GitHub Actions primarily rely on logs and 2D dashboards that present information on a per-run basis. These representations offer limited support for visualizing temporal patterns, concurrency, or failure propagation, making it harder for users to detect regressions, understand dependencies, or reason about performance issues across multiple runs [39]. Addressing

these challenges requires new approaches that align with how users perceive time and structure. One such approach is immersive analytics, which enhances understanding through spatial layout and embodied interaction. Although effective in other domains, it remains largely unexplored in CI/CD workflows.

This dissertation addresses the lack of practical tools for intuitive, interactive, and immersive exploration of CI/CD workflows. It investigates how virtual reality can be used to represent workflow executions in a spatially meaningful way, supporting more explicit temporal reasoning, structural navigation, and failure diagnosis.

3.2 CI/CD Execution Model

CI/CD pipelines in GitHub Actions are organized around a structured hierarchy that controls how automated tasks are defined and executed. This hierarchy includes three key elements: workflows, workflow runs, and jobs. Understanding their semantics is essential for modeling these pipelines visually.

Workflow: A static YAML file committed to a repository, defining a set of jobs and their execution logic. Each workflow is triggered by specific events, such as pushes or pull requests.

Workflow Run: A dynamic execution instance of a workflow created in response to a triggering event. Workflow runs are characterized by start and end times, execution outcomes, and instances of job executions.

Job: A unit of work executed within a workflow run. Jobs consist of sequential steps and may run concurrently or depend on the completion of other jobs using the `needs` keyword.

Steps and Actions: Jobs are composed of sequential steps that execute shell commands, scripts, or reusable actions

To illustrate these concepts, consider the example workflow YAML file shown in Listing 3.1:

Table 3.1: Core CI/CD elements and key attributes.

Element	Key Attributes	Description
Workflow	Name, Trigger type	Static definition of job structure and execution logic.
Workflow Run	Run ID, Status, Start/End time, Outcome	Instance of workflow execution triggered by an event.
Job	Name, Status, Duration, Dependencies	Executable unit within a run, may depend on or run in parallel with other jobs.
Steps	Step name, Execution order, Action reference	Ordered commands or actions within a job, such as shell scripts, tests, or tool invocations.

```
1 name: CI Pipeline                                # Workflow
2 on: [push, pull_request]                        # Trigger events
3
4 jobs:
5   test:                                          # Job 1
6     runs-on: ubuntu-latest
7     steps:                                     # Steps within the job
8       - uses: actions/checkout@v3             # Action step
9       - name: Run tests                        # Command step
10        run: npm test
11
12   build:                                       # Job 2
13     needs: test                               # Dependency on 'test' job
14     runs-on: ubuntu-latest
15     steps:
16       - uses: actions/checkout@v3
17       - name: Build application
18        run: npm run build
19
20   deploy:                                     # Job 3
21     needs: [test, build]                     # Dependencies on multiple jobs
22     runs-on: ubuntu-latest
23     if: github.ref == 'refs/heads/main'
24     steps:
25       - uses: actions/checkout@v3
26       - name: Deploy to production
27        run: ./deploy.sh
```

Listing 3.1: Example GitHub Actions workflow showing the hierarchical structure

These elements form a clear hierarchical and temporal model: workflows define job structures; workflow runs execute them in response to events; and jobs are scheduled within runs based on their declared dependencies. Jobs without dependencies execute in parallel, while dependent ones follow a defined order. Multiple workflow runs may also execute concurrently if triggered in rapid succession.

Figure 3.1 illustrates the relationship between these elements, highlighting both containment and execution dependencies as they appear in the example workflow.

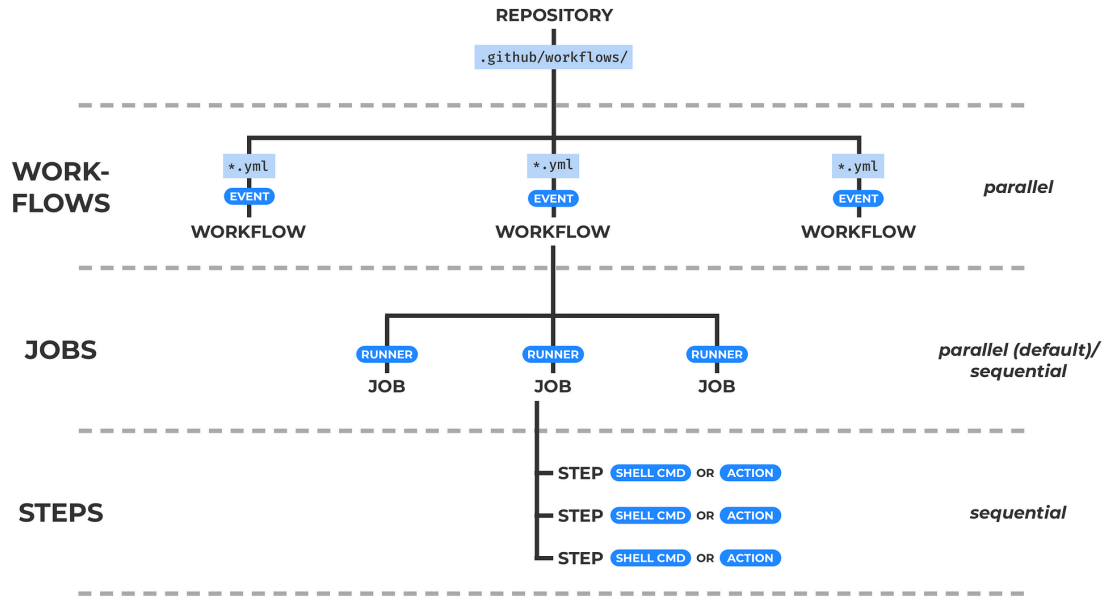


Figure 3.1: Hierarchy and dependency structure in GitHub Actions workflows, showing the relationship between the workflow definition (Listing 3.1) and its execution instances

This structural and temporal model guides the spatial layout and encoding strategies discussed later in this chapter.

3.3 User Tasks and Visualization Requirements

To create an effective visualization for CI/CD pipelines, it is essential to understand the core reasoning tasks developers perform when interacting with these systems. Prior research has highlighted key challenges in maintaining, debugging, and optimizing CI/CD workflows.

Common Developer Tasks

- **Failure Detection and Diagnosis:** Locating failures and understanding their causes is critical to maintaining reliability. Users need to trace failing jobs and identify patterns or propagation across workflows.
- **Temporal Pattern Recognition:** Developers benefit from seeing how workflow behavior evolves over time to identify regressions, recurring issues, or intermittent problems.
- **Concurrency and Dependency Inspection:** Understanding how jobs run in parallel or sequence is essential for debugging timing-related issues and optimizing performance.
- **Comparative Analysis:** Comparing runs across different commits or branches helps in tracking regressions or evaluating the impact of changes.

- **Navigation and Drill-down:** Users must move fluidly between high-level views of the pipeline and detailed information about specific runs and jobs.

Supporting Tasks Through Visualization

These tasks relate directly to the CI/CD elements defined earlier. Table 3.2 summarizes the relevant elements and attributes for each task.

Table 3.2: Developer tasks and corresponding CI/CD elements.

Task	Relevant Elements	Key Attributes
Failure Detection and Diagnosis	Workflow Runs, Jobs	Outcome, Duration, Dependencies
Temporal Pattern Recognition	Workflow Runs	Start/End time, Trigger event
Concurrency and Dependency Inspection	Jobs	Dependencies, Execution order
Comparative Analysis	Workflow Runs, Workflows	Trigger, Attempt number, Outcome
Navigation	Workflow Runs, Jobs	Hierarchical structure, Timing

Visualization and Interaction Goals

From the identified tasks, we derive the following goals for the visualization model:

1. **Temporal Clarity:** Clearly reveal execution timing, sequencing, and overlaps.
2. **Hierarchical Representation:** Make workflow, run, and job structure visually explicit.
3. **Failure and Status Highlighting:** Use visual and auditory cues to distinguish outcomes and prioritize failed runs.
4. **Comparative Context:** Allow users to compare multiple runs effectively.
5. **Scalability and Focus:** Support both broad overviews and fine-grained inspection without overwhelming users.
6. **Multimodal Feedback:** Utilize visual and auditory channels to emphasize important information and effectively direct the user's attention.

These goals inform the design choices discussed in Sections 3.4, 3.5 and 3.6, grounding them in real-world user needs.

3.4 Multimodal Design Considerations

Effectively visualizing complex CI/CD pipelines can demand using multiple perceptual channels to communicate various types of information efficiently. While visual encoding handles structural

and temporal relationships, auditory feedback can be a complementary channel for critical events and status changes.

Auditory Design Principles

Auditory cues follow established principles for non-intrusive but informative feedback:

- **Event-Driven Sounds:** Distinct audio signals associated with specific pipeline events (e.g., runs or job failures).
- **Spatial Audio:** Positional audio corresponding to the 3D layout allows users to locate events without direct visual inspection.
- **Severity Mapping:** Different sound characteristics (pitch, intensity, duration) that reflect the criticality of events, with failures receiving more prominent auditory treatment.

This multimodal approach ensures that users can monitor pipeline health through ambient awareness while focusing their visual attention on detailed analysis tasks.

3.5 Design Metaphor

To support an intuitive understanding of CI/CD pipelines, a spatial and metaphorical model was developed that maps the abstract execution semantics of GitHub Actions into a physical structure. The chosen metaphor is inspired by the layout of a *multi-story industrial building*, where each floor represents a different workflow, and workflow runs and jobs are arranged within and across those floors to reflect their execution over time.

This metaphor is intended to provide a stable mental model that reinforces the temporal, hierarchical, and concurrent nature of CI/CD systems while remaining general enough to be adapted into different visualization formats. Each visual element corresponds to a fundamental concept in GitHub Actions, supporting clarity and traceability throughout the visualization.

Mapping Core Elements to the Metaphor

The spatial metaphor maps each CI/CD element to a distinct representation, as illustrated in Figure 3.2:

- **Workflow** → *Floors of the building*. Each workflow defined in the repository is assigned a distinct horizontal layer, vertically stacked with others. This vertical separation reinforces the independence and parallelism of workflows while allowing comparison across them.
- **Workflow Run** → *Units arranged across and along the depth axis (X- and Z-axis) on each floor*. Each workflow run appears as a discrete instance positioned based on when it was triggered and whether it overlaps in time with other runs. Runs that execute concurrently are placed side by side, while sequential runs are positioned one after the other in depth.

- **Job** → *Subcomponents inside each workflow run unit.* Jobs are arranged laterally within their respective run container. As with workflow runs, parallel jobs are displayed side by side, while jobs constrained by dependencies are positioned sequentially.

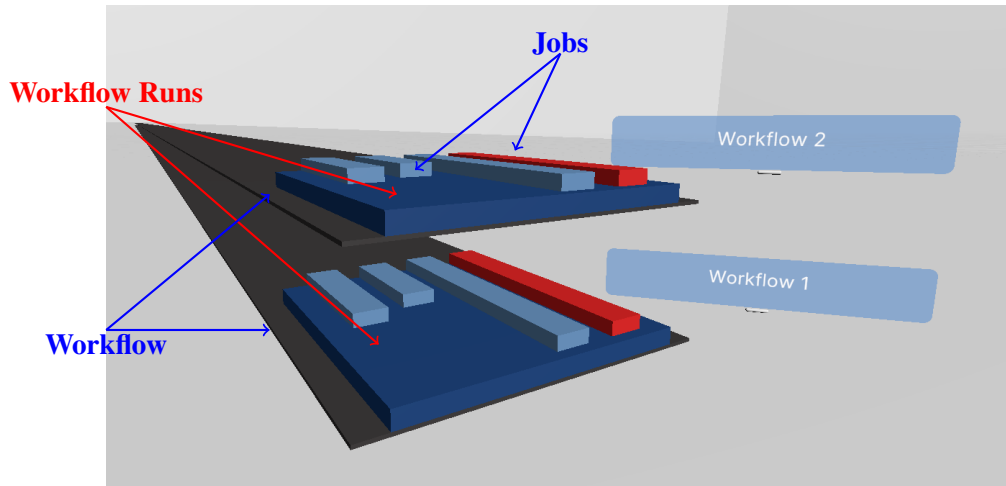


Figure 3.2: Conceptual design metaphor showing the mapping of CI/CD elements to spatial representations in the building-inspired layout

Supporting Temporal and Structural Reasoning

This metaphor serves multiple cognitive functions aligned with the visualization goals outlined earlier:

- **Hierarchy:** The vertical stacking of floors separates different workflows, while the containment of jobs inside workflow runs reflects the execution hierarchy. Users can distinguish between the structural levels at a glance.
- **Time:** Temporal sequencing is expressed through spatial progression along the Z-axis. This allows users to intuitively read time as depth, identifying when runs occurred and observing temporal proximity between them.
- **Concurrency:** Both workflow runs, and jobs that execute concurrently are displayed side by side. This arrangement highlights overlaps in execution at multiple levels, making parallel activity immediately visible and aiding the diagnosis of concurrency-related issues.
- **Failures and Comparison:** Because workflow runs are displayed as aligned units across time and workflows, users can more easily compare job outcomes across runs and identify patterns in pipeline behavior over time.

By adopting this building-inspired metaphor, the design provides a stable spatial framework that supports exploration, comparison, and analysis of CI/CD pipelines, establishing the foundational logic upon which later visual and interactive representations are based.

3.6 Visual Encoding and Structural Mapping

With the metaphor established, this section clarifies the spatial arrangement and visual representation of core CI/CD elements. These representations establish the structural rules that dictate the layout, grouping, and appearance of elements within the visualization.

Spatial Dimensions and Layout

The three-dimensional layout uses space to represent structural hierarchy and temporal relationships:

- **Z-axis – Temporal Progression:** Time flows along the depth axis. More recent workflow runs are displayed at the front, while older runs move towards the back. Similarly, within each workflow run, jobs are sequenced along the Z-axis according to their execution order, with earlier jobs positioned towards the back and later jobs towards the front.
- **X-axis – Parallelism and Grouping:** Lateral space is used to display concurrency. Workflow runs that execute concurrently are placed side by side along the X-axis. Within each run, jobs that start simultaneously or in parallel are positioned next to each other, making execution overlaps immediately visible.
- **Y-axis – Workflow Layering:** The vertical axis separates workflows into distinct layers or floors. Each floor is reserved for a single workflow definition, which keeps execution units organized and facilitates comparison across workflows without visual interference.

Grouping and Containment

Logical grouping reinforces structural clarity:

- **Jobs within Workflow Runs:** All jobs in a single workflow run are within a shared bounding space. This containment reinforces their shared execution context and allows users to treat a run as a coherent analytical unit.
- **Runs within Workflows:** Workflow runs are aligned sequentially along the Z-axis and grouped within the same vertical layer. This preserves the relationship between a workflow definition and its executions while allowing temporal and structural reasoning.
- **Separation of Workflows:** By assigning each workflow its own floor, the design avoids visual entanglement between unrelated executions and supports more explicit inter-workflow comparisons.

Visual Attributes for State and Status

While layout encodes structural and temporal properties, additional visual attributes are used to represent the execution state:

- **Color – Execution Outcome:** Color is used to differentiate successful, failed, or cancelled jobs and runs. These colors provide immediate feedback about pipeline health and help direct user attention toward anomalies or failures.
- **Shape and Size – Abstraction Level:** Workflow runs, and jobs are represented with simplified geometric shapes (e.g., blocks or units) sized consistently for comparability. Visual uniformity supports pattern recognition and avoids distortion of relative importance.
- **Spatial Gaps – Execution Gaps:** Gaps between runs or jobs may represent idle periods, execution delays, or asynchronous activity, further reinforcing temporal reasoning.

Encoding Principles Summary

The visual encoding strategy prioritizes consistency, legibility, and structural clarity. Each design choice is grounded in a semantic mapping (Table 3.3).

Table 3.3: Visual encoding mapping between CI/CD concepts and visual representations

CI/CD Concept	Visual Mapping	Purpose
Workflow	Vertical floor	Separate pipelines, support comparison
Workflow Run	Unit along Z-axis (time), placed laterally if concurrent	Show temporal order and concurrency
Job	Element inside run unit laid out laterally	Show structure and dependencies
Execution Outcome	Color (e.g., green, red, gray)	Indicate success, failure, cancellation
Execution Time	Z-position and optional gaps	Represent when and how long executions occurred

The visualization model provides a foundation for later interaction and immersion by systematically encoding structure, timing, and status. These mappings serve as the visual grammar through which CI/CD pipelines are interpreted and explored without relying on traditional 2D charts or textual logs.

3.7 Interaction Design Considerations

While the previous sections outlined the structural and visual encoding of CI/CD workflows, a practical analytic experience also depends on the user’s ability to dynamically explore, compare, and reason about the system. This section identifies the interaction goals necessary to support meaningful engagement with the visualization.

Navigating and Exploring Workflow Space

Given the multi-dimensional layout of the visualization, users must be able to traverse the spatial model effectively. This includes the ability to:

- Move across different workflow floors to examine how distinct pipelines behave.
- Progress along the temporal axis to review past or recent executions.
- Focus on specific runs or jobs, adjusting viewpoint or level of detail to support inspection.

These navigation goals ensure that the system supports both broad overviews and targeted analysis.

Inspecting and Comparing Execution Units

Users require access to metadata and contextual information for specific runs and jobs to support deeper analysis. This involves:

- Selecting individual workflow runs to retrieve execution status, timing, trigger context, and other metadata.
- Inspecting jobs within a run to understand their order, duration, and outcome.
- Comparing two or more workflow runs from the same or different workflows to identify behavioral changes or recurring patterns.

These interactions support debugging, regression tracking, and validation of pipeline behavior across different commits or branches.

3.8 Summary

This chapter presented the conceptual framework for immersive CI/CD visualization, including the core execution model, spatial metaphor, visual encoding, and interaction design. These elements form the foundation for translating conceptual structure into an interactive system, which is addressed in the next chapter.

Chapter 4

Prototype Implementation

This chapter presents the technical implementation of the prototype developed to support the research objectives outlined in previous chapters. The system transforms GitHub Actions workflow data into an interactive virtual reality (VR) environment, enabling users to explore the structure, execution, and temporal dynamics of CI/CD pipelines through spatial, visual, and auditory cues. Its primary purpose is to investigate how immersive analytics techniques can enhance understanding of software automation processes compared to traditional two-dimensional tools.

The prototype was designed to support a user study comparing VR-based and conventional CI/CD activity representations. To achieve this, it implements a spatial metaphor based on a multi-level industrial factory, where workflows, runs, and jobs are distributed across vertical and horizontal axes according to their hierarchy and execution order. Time is encoded along the depth of the environment, allowing users to navigate backward through workflow history in a physically embodied manner.

The system was built using Unity¹ and the XR Interaction Toolkit² and is compatible with various VR configurations, including seated, standing, and room-scale setups. Workflow data is obtained from the GitHub Actions platform, either through live API queries or from preprocessed datasets that are stored locally. The prototype incorporates interactive tools, accessibility features, and performance optimizations to ensure both usability and experimental reliability.

4.1 System Architecture

The prototype follows a layered architecture designed to support extensibility, maintainability, and the performance constraints of immersive virtual reality. It is structured into four primary subsystems: data handling, visualization, interaction, and audio. Each subsystem is responsible for a distinct aspect of the system's behavior. These subsystems collectively transform GitHub Actions data structures, namely workflows, runs, and jobs, into a spatial, interactive representation within a 3D environment.

¹Unity Game Engine: <https://unity.com/>

²Unity XR Interaction Toolkit Documentation: <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html>

The system is implemented in Unity using C# and relies on the XR Interaction Toolkit for cross-platform VR support. Rendering is handled through the Universal Render Pipeline (URP)³, with TextMeshPro⁴ used for sharp text rendering. Unity’s Input System ensures consistent interaction mapping across different VR devices. Table 4.1 summarizes the main technologies used in the implementation.

Figure 4.1 illustrates the overall system architecture, highlighting how the four subsystems interact to support immersive CI/CD pipeline visualization.

Table 4.1: Key technologies used in the implementation.

Technology	Version	Purpose
Unity 3D	6000.0.24f1	Core engine and scene management
C# (.NET)	Standard 2.1	Scripting language
XR Interaction Toolkit	3.0.6	VR interaction abstraction
Universal Render Pipeline	17.0.3	Optimized rendering for VR
TextMeshPro	3.0+	High-quality text rendering
Unity Input System	1.11.1	Cross-device input support

Subsystems Overview

The architecture is composed of four distinct subsystems, each handling a specific aspect of the immersive visualization experience. These subsystems work together to transform raw GitHub Actions data into an interactive 3D environment while maintaining a clear separation of concerns for maintainability and extensibility.

Data Handling: Manages the retrieval, parsing, and internal representation of GitHub Actions workflow data. It supports live API access and offline JSON datasets through a provider pattern, abstracting data sources behind a common interface (detailed in Section 4.2).

Visualization: Responsible for spatializing workflows, runs, and jobs using the factory metaphor. It includes algorithms for timeline scaling, layout placement, and temporal compression. This subsystem also implements the visual encoding of run and job statuses using color-coded materials. For accessibility, it includes a colorblind-safe palette (detailed in Section 4.6).

Interaction: Supports user navigation and interaction with the 3D environment. This includes multiple locomotion options (joystick, teleportation, physical movement), vertical traversal between workflow floors, and hover-based inspection panels for detailed run metadata. Features for accessibility are included in this subsystem, such as toggles to turn on or off audio cues and color modes (detailed in Section 4.5).

Audio: Provides spatialized sound cues for failed runs, helping users locate issues using sound. Audio is triggered based on workflow status and positioned accurately in the 3D space, complementing visual feedback (detailed in Section 4.4).

³Unity Universal Render Pipeline: <https://unity.com/features/srp/universal-render-pipeline>

⁴Unity TextMeshPro Documentation: <https://docs.unity3d.com/Packages/com.unity.ugui@3.0/manual/TextMeshPro/index.html>

Modularity and Extensibility

The system architecture emphasizes modularity. Data is abstracted using a provider pattern, supporting live API fetching and static file loading without altering downstream logic. Visual elements are instantiated using prefabs, allowing for efficient reuse and scalable layout generation. Materials are loaded dynamically from the `UnityResources` folder at runtime, making it easy to modify or extend the visual encoding scheme without recompiling the application.

Configuration values for layout spacing, time scaling, and interface behavior are exposed through the Unity Inspector, enabling fine-tuning during testing. The separation of concerns across components supports independent testing and facilitates future extensions, such as supporting other CI/CD platforms or expanding the range of interaction techniques.

4.2 Data Layer

The data layer serves as the basis of the VR system by simplifying the retrieval, transformation, and structuring of GitHub Actions data into a format suitable for immersive visualization. It supports real-time data acquisition from the GitHub REST API and the loading of preprocessed static JSON dataset files. Internally, all data is structured to match the hierarchy of GitHub Actions: workflows contain multiple runs, and each Run contains multiple jobs.

Data Sources and Repository Selection

The system supports two data sources:

- **GitHub REST API:** Real-time CI/CD data is fetched using asynchronous HTTP requests through Unity's `UnityWebRequest` system. API access supports public and authenticated modes, with personal access tokens managed via a configuration file.
- **Locally stored JSON files:** Static datasets are bundled within Unity's `Resources` folder and include real snapshots and synthetic datasets already transformed into the internal format. These allow consistent offline testing with predictable and repeatable test conditions without further preprocessing.

API Endpoints Used

Data from GitHub Actions API is retrieved in a three-stage hierarchical process using the REST API endpoints shown in Table 4.2.

Each endpoint supports pagination and optional date filtering. For the workflow runs endpoint, date constraints are added using `created` query parameters (e.g., `created=>2024-09-01`). These filters help reduce the retrieved data volume, which is crucial for performance in immersive contexts. On the client side, additional boundary validation ensures that only runs within the specified range are processed. Partial ranges (e.g., "after date X") are also supported.

Table 4.2: GitHub REST API endpoints used in the system.

Note: All endpoints use the base URL pattern `/repos/{owner}/{repo}/actions/...`

Endpoint	Purpose
<code>.../workflows</code>	Lists all workflows in the target repository and retrieves their metadata
<code>.../workflows/{id}/runs</code>	Retrieves runs associated with a specific workflow, with support for date-based filtering
<code>.../runs/{id}/jobs</code>	Retrieves all jobs associated with a specific workflow run

Provider Pattern and Implementation

To separate data retrieval logic from the rest of the system, the application uses a provider pattern based on a shared interface:

```
public interface IWorkflowDataProvider {
    Task<WorkflowData> GetWorkflowDataAsync();
}
```

Two implementations of this interface are available:

- `GitHubApiDataProvider` fetches data directly from the GitHub REST API. It supports pagination, date filtering, and authentication handling. Data is retrieved in three stages: workflows, runs per Workflow, and jobs per Run.
- `LocalJsonDataProvider` loads data from JSON files bundled with the application. This enables consistent results during development and testing and supports the use of both real-world and synthetic datasets.

Both providers deliver the data using the same internal structure. The system uses coroutine-based loading to ensure non-blocking execution and paginates API requests to manage memory usage. Visual objects are instantiated only after data validation, avoiding unnecessary overhead and ensuring frame rate stability in VR.

Date Filtering and Temporal Data Management

The system implements date filtering capabilities to manage data volume and focus visualization on specific periods:

- **API-Level Filtering:** GitHub API requests include `created` query parameters to filter workflow runs at the source, reducing network overhead and processing time.
- **Client-Side Refinement:** Additional date boundary validation ensures precise temporal filtering, handling edge cases where API filtering may be approximate.

- **Configurable Date Ranges:** The system supports flexible date range specifications with optional start and end dates, enabling partial range filtering (e.g., "from date X" or "until date Y").
- **Date Range Validation:** Date parsing ensures consistent interpretation of input formats and temporal alignment across datasets.

Date filtering is particularly valuable for large repositories where the complete history would exceed VR performance constraints, allowing users to focus on specific development periods or recent activity patterns.

Data Transformation and Validation

The system transforms GitHub API responses into internal serializable classes that align with Unity's serialization model. The transformation is hierarchical: each Workflow contains a list of Runs, and each Run contains a list of Jobs. During this process:

- All timestamps (e.g., `created_at`, `started_at`) are parsed for timeline positioning.
- Parent-child relationships are validated to ensure structural consistency.
- Optional fields are handled using default values or fallback substitution to ensure data integrity and prevent runtime exceptions (e.g., using run-level timestamps when job-level data is unavailable)

Error Handling and API Constraints

The system implements robust error handling to manage GitHub API constraints:

- **Rate Limiting:** API requests include proper authentication headers and implement pagination (100 items per page) to stay within GitHub's rate limits.
- **Network Resilience:** Failed requests are logged with detailed error messages.

Resulting Data Structure Format

After transformation, the data is loaded into the system in a unified structure suitable for visualization. The simplified format in Listing 4.1 illustrates the internal model:

This structure allows direct mapping to spatial and visual elements in the 3D scene, preserving hierarchy, execution order, and status.

Real and Synthetic Dataset Support

During development, the system primarily relied on synthetic datasets inspired by the Facebook React repository's CI/CD structure. These JSON-based synthetic datasets simulate realistic CI/CD patterns while enabling controlled testing with known properties, such as:

```

1 {
2   "workflows": [
3     {
4       "id": 12345,
5       "name": "CI",
6       "state": "active",
7       "path": ".github/workflows/ci.yml",
8       "runs": [
9         {
10          "id": 56789,
11          "status": "completed",
12          "conclusion": "success",
13          "event": "push",
14          "created_at": "2024-12-01T10:00:00Z",
15          "jobs_list": [
16            {
17              "id": 111,
18              "name": "Build",
19              "status": "completed",
20              "conclusion": "success",
21              "started_at": "2024-12-01T10:01:00Z",
22              "completed_at": "2024-12-01T10:05:00Z"
23            },
24            {
25              "id": 112,
26              "name": "Test",
27              "status": "completed",
28              "conclusion": "failure",
29              "started_at": "2024-12-01T10:05:10Z",
30              "completed_at": "2024-12-01T10:08:30Z"
31            }
32          ]
33        }
34      ]
35    }
36  ]
37 }

```

Listing 4.1: Unified data structure format for visualization

- Varied success and failure distributions
- Simultaneous job execution across workflows
- Extended or minimal durations for timeline scaling
- Predictable temporal patterns for algorithm validation

The synthetic approach allowed for rapid prototyping and feature development without the constraints of API rate limits or network dependencies. Both synthetic and captured datasets from GitHub repositories are stored in JSON format and processed identically through the local JSON data provider (`LocalJsonDataProvider`), ensuring consistent data handling regardless of the data source origin. The selection between live API data, synthetic datasets, and cached snapshots is configurable before starting the system.

4.3 Factory Visualization System

This section describes the implementation of the spatial factory metaphor used to visualize GitHub Actions workflows in the VR system. The layout is hierarchically structured, with each level of the CI/CD pipeline mapped to a corresponding visual component: floors for workflows, conveyor belts for timelines, blocks for workflow runs, and sub-blocks for jobs. All elements are procedurally instantiated and spatially arranged based on the underlying workflow data and a shared temporal reference system.

Figure 4.2 provides an overall view of the implemented factory layout. It illustrates the complete hierarchical structure, which includes multiple workflow floors, conveyor belts, and workflow runs positioned according to their temporal execution. Additionally, it features temporal timeline markers, temporal portals (to allow collapsing long periods of time without events), and all other elements present in the system that will be discussed in the following sections.

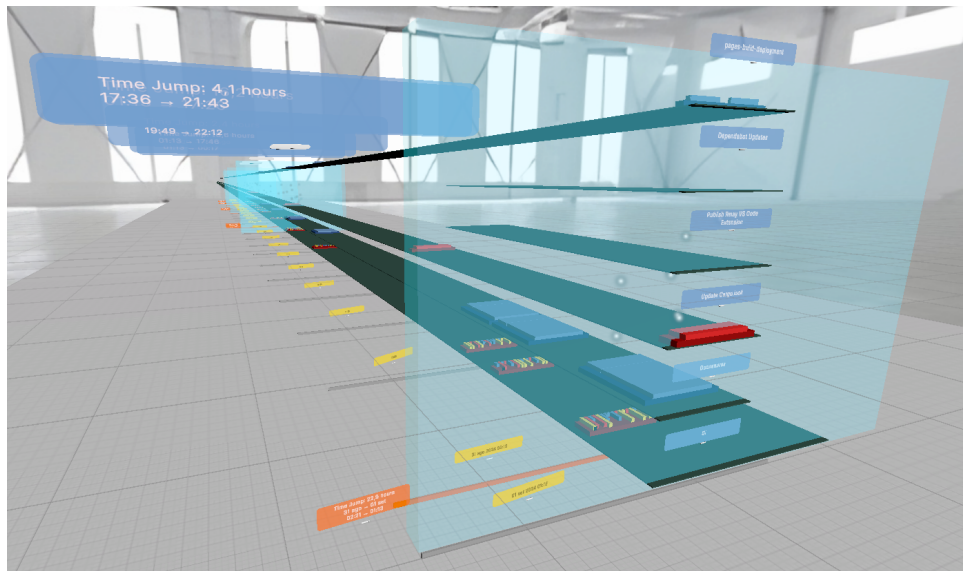


Figure 4.2: Complete factory layout demonstrating the full visualization system implementation. The image shows multiple workflow floors stacked vertically (each at 3.5-unit intervals), conveyor belts spanning workflow timelines, positioned workflow runs as blocks, jobs as sub-blocks, and temporal markers featuring timestamps and temporal portals. This view illustrates the integration of all spatial factory metaphor components working together to visualize GitHub Actions CI/CD data.

Temporal Foundation and Reference System

Before establishing the spatial layout, the system creates a temporal foundation that translates workflow timestamps into spatial positions along the Z-axis, turning time into a navigable dimension within the immersive environment. All positioning calculations operate relative to a shared global reference point and a configurable time scale.

Z-Axis as Time Axis:

Time is encoded along the Z-axis, from the most recent events near the user ($Z = 0$) to older events extending further away ($Z+$). This temporal layout places the present closest to the entry point and the past further into the scene, enabling intuitive backward exploration through CI/CD history.

The reference point is computed by collecting all timestamps (start and end) from every workflow run and selecting the latest one. This reference time defines the temporal origin:

$$z = (\text{referenceTime} - \text{eventTime}) \times \text{timeScale}$$

Formula Components:

- **z**: Final Z-coordinate position along the temporal axis
- **referenceTime**: Latest timestamp across all workflows in the dataset
- **eventTime**: Timestamp of the specific workflow run or job being positioned
- **timeScale**: Conversion factor set to 0.5 world units per minute for temporal-to-spatial mapping

This setup ensures that newer runs are always placed closer to the user and older runs consistently further away, preserving temporal context throughout the factory environment.

Timeline Markers and Navigation Aids

Timeline markers are instantiated at 30-minute intervals along the factory floor to support temporal orientation. These are visualized as thin vertical structures anchored to the left side of the scene ($X = -5$) and extending across the temporal depth of the environment. Their labels are positioned at $X = -10$ and $Y = 1.5$ for legibility as shown in Figure 4.3.

Timeline markers follow an adaptive labeling strategy:

- Full format (dd MMM yyyy HH:mm) is used for the first marker, day transitions, and after time gaps
- Compact format (HH:mm) is used for regular intervals
- Markers are gray by default and turn orange after time jumps to indicate a resumption of activity (as detailed in Section 4.3)

This visual rhythm supports user navigation by anchoring workflow activity to absolute time references, aiding in both detailed inspection and macro-level interpretation of CI/CD patterns.

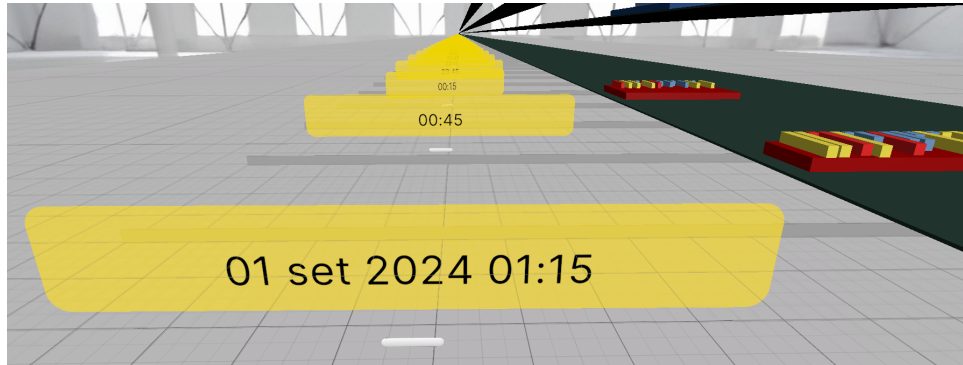


Figure 4.3: Timeline markers positioned at 30-minute intervals along the factory floor, providing temporal reference points for navigation and orientation within the VR environment.

Workflows as floors

Each GitHub workflow is represented as a horizontal floor in the 3D scene. Floors are vertically stacked from ground level upward using a constant spacing of 3.5 units, with workflow positioning determined by their order in the loaded dataset. The first workflow in the dataset is positioned at the base height (1.0 units above ground level), with subsequent workflows stacked vertically above it. This positioning is computed using the formula:

$$y = baseHeight + (workflowIndex \cdot ySpacing)$$

where `workflowIndex` corresponds directly to the workflow's position in the original dataset (0 for the first workflow, 1 for the second, etc.).

Figure 4.4 demonstrates the practical application of the positioning formula.

Formula Components:

- **y**: The final vertical position (Y-coordinate) for the workflow label reference point
- **baseHeight**: Set to 1.0 world units as the reference height for workflow labels, with other elements positioned relative to this baseline
- **workflowIndex**: The zero-based index of the workflow in the processing loop, corresponding to the workflow's order in the loaded dataset (0 for first workflow, 1 for second, etc.)
- **ySpacing**: Fixed spacing of 3.5 world units between consecutive workflow floors, providing adequate vertical separation for VR navigation

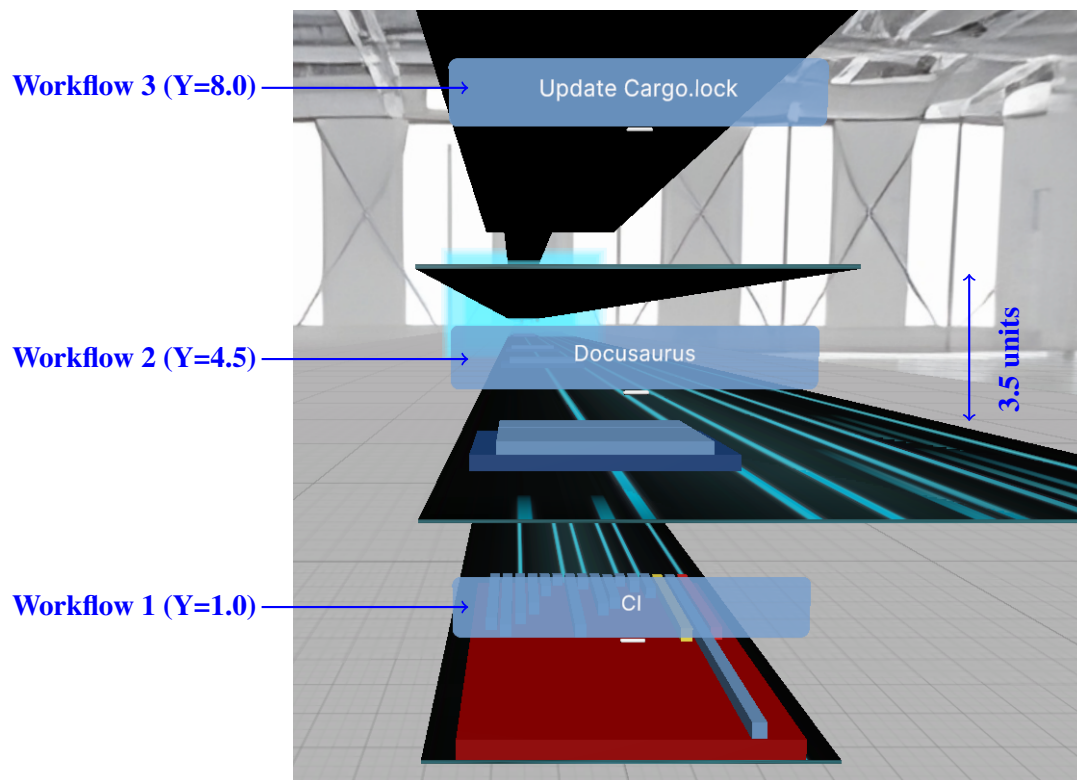


Figure 4.4: Three workflow floors showing vertical stacking with labels: "CI" (bottom), "Docusaurus" (middle), and "Update Cargo.lock" (top), demonstrating the 3.5-unit spacing between floors.

This creates evenly spaced floors with adequate vertical separation while keeping the first workflow clearly above the ground level.

For each workflow, a physical floor is instantiated and automatically scaled to match the length of the workflow's temporal timeline. The timeline represents the complete temporal span from the earliest workflow run to the latest activity within that specific workflow. All workflow elements are hierarchically organized under a parent container to maintain a structured scene arrangement and enable efficient operations.

Figure 4.5 provides additional perspective views of the workflow floor arrangement, highlighting the spatial organization and scale relationships in the VR environment.

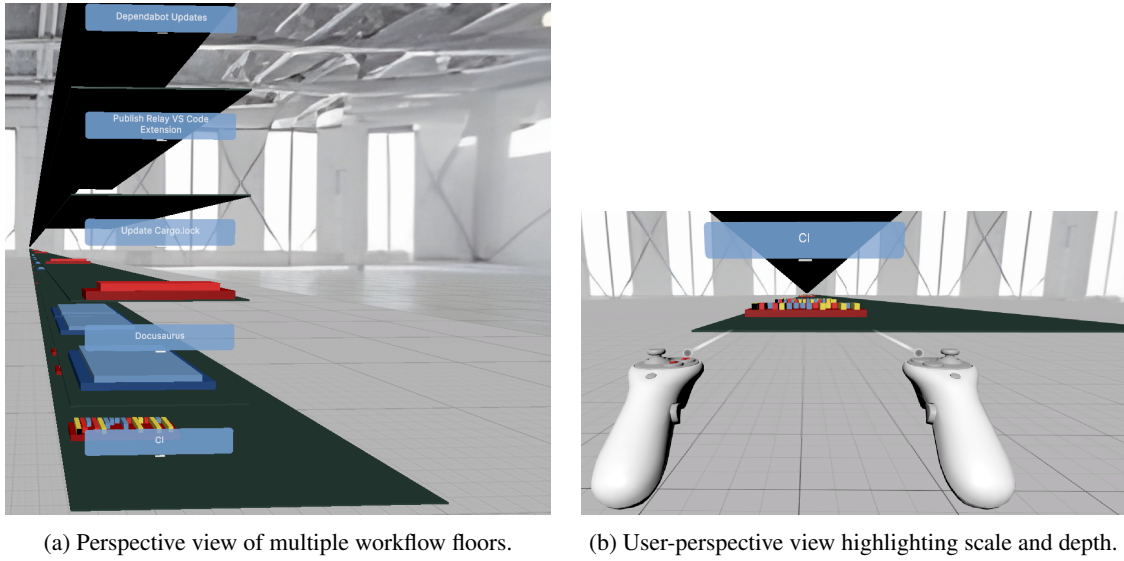


Figure 4.5: Workflow floor spatial arrangement from different perspectives, demonstrating the vertical stacking system and VR environment scale.

Conveyor Belts as Workflows Temporal Infrastructure

Each workflow floor includes a conveyor belt that provides the temporal foundation for workflow runs. The conveyor belt spans the complete timeline of the workflow and adapts dynamically to accommodate all runs within that workflow:

- **Width Adaptation:** The belt width encompasses all workflow runs horizontally. The system calculates the leftmost and rightmost run positions, adds 0.5-unit margins, and centers the belt at the midpoint between these bounds.
- **Length Adaptation:** The belt length represents the workflow's complete temporal span, calculated using:

$$length = (referenceTime - earliestTimestamp) \cdot timeScale + margin$$

Component	Description
referenceTime	Latest timestamp across all workflows in the dataset, serving as the temporal reference point
earliestTimestamp	Earliest <code>run_started_at</code> within the specific workflow being processed
timeScale	Conversion factor set to 0.5 world units per minute for temporal-to-spatial mapping
margin	Fixed 10-unit buffer added for visual separation and timeline clarity

Table 4.3: Conveyor belt length calculation components

The conveyor belt is positioned 0.6 units below the workflow label reference point, which places it approximately 0.4 units above the actual floor level (since floors are positioned 1.0 unit below the label reference). This 0.4-unit elevation resembles real-world conveyor belt heights and provides comfortable viewing and interaction angles for VR users. A default length of 25 units is used when no valid timestamps are available.

Figure 4.6 shows an empty conveyor belt above a workflow floor.

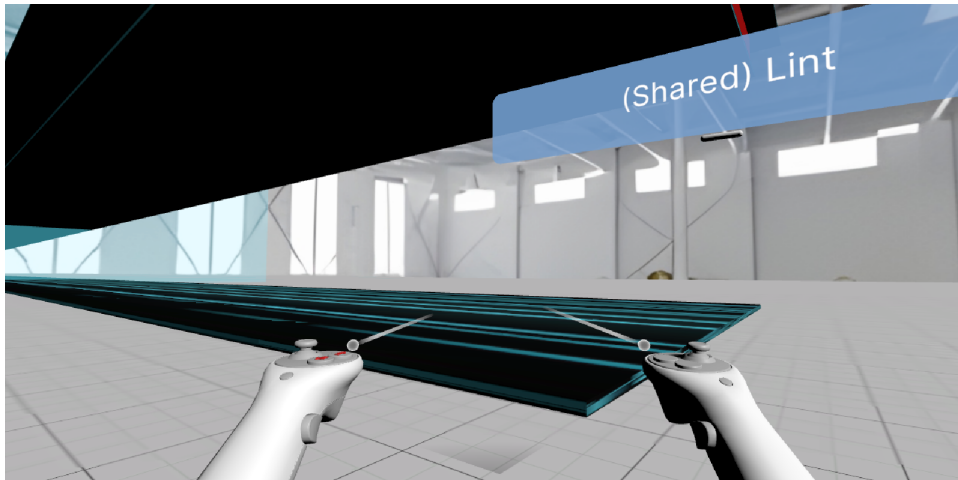


Figure 4.6: Empty conveyor belt on a workflow floor, positioned above ground level.

Workflow Runs as Blocks

Workflow runs are represented as rectangular blocks positioned on the conveyor belt. Each run consists of a base block encoding its temporal duration and execution status, with smaller job blocks positioned within it to represent individual tasks. Figure 4.7 demonstrates a single workflow run with its job blocks.

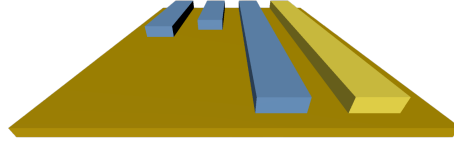


Figure 4.7: A single workflow run visualized as a base block containing job blocks. The base block represents the run's temporal span while job blocks show individual task execution.

Positioning System:

Workflow runs are positioned using both temporal and spatial coordinates:

- **Temporal Positioning (Z-axis):** Run position along the timeline is calculated based on execution timestamps:

$$zPosition = zStart + \frac{length}{2}$$

where $zStart$ is derived from the run start time and $length$ represents the run duration.

- **Spatial Positioning (X-axis):** Parallel runs are arranged using a lane assignment algorithm to prevent overlaps:

$$x = laneIndex \cdot (runWidth + laneGap)$$

Formula Components:

- **x:** Final horizontal position (X-coordinate) for the workflow run block
- **laneIndex:** Zero-based lane number assigned by the algorithm (0 for first lane, 1 for second, etc.)
- **runWidth:** Width of the workflow run block (derived from the prefab dimensions)
- **laneGap:** Default spacing of 2 units between adjacent lanes to ensure visual separation

Lane Assignment Algorithm:

The system uses a Greedy lane assignment approach:

1. Runs are sorted chronologically by start time
2. Each run is assigned to the first available lane where no temporal overlap occurs
3. New lanes are created when overlaps are detected with existing runs

Temporal overlap is determined using the condition:

$$startTime_A < endTime_B \quad \text{and} \quad startTime_B < endTime_A$$

Workflow runs are positioned on top of the conveyor belt infrastructure and scaled proportionally to their temporal duration. Runs materials indicate execution status through color coding (success, failure, in progress).

Figure 4.8 demonstrates the visual status encoding for different workflow execution states, showing the color-coding system that will be detailed in Section 4.6.

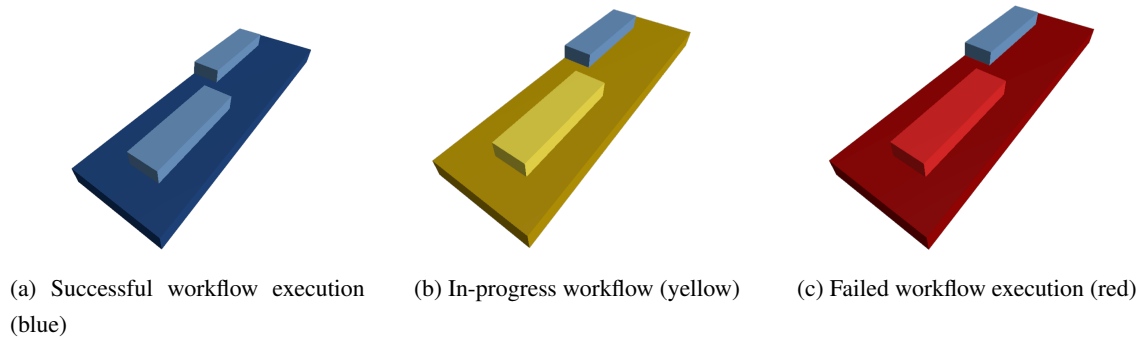


Figure 4.8: Workflow run status visualization through color coding: success, in-progress, and failure states.

Figure 4.9 illustrates different run arrangements, showing how the lane assignment algorithm handles parallel and sequential execution patterns.

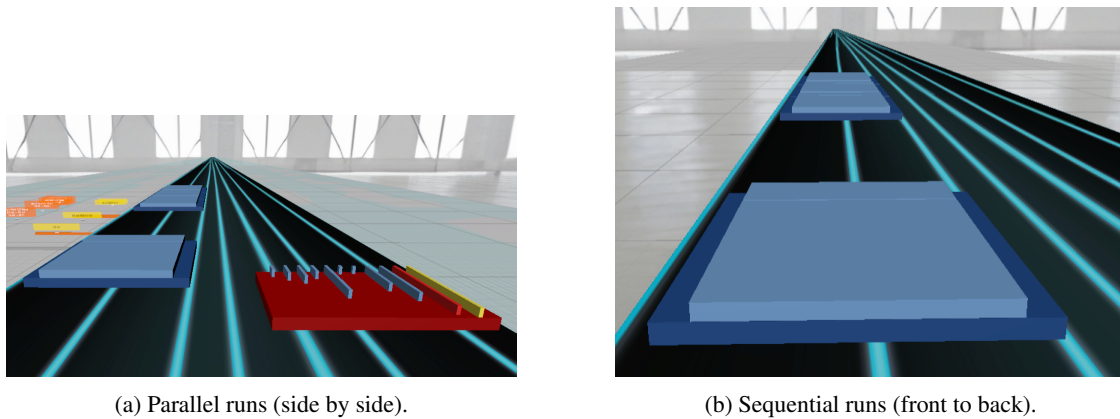


Figure 4.9: Spatial encoding of workflow runs: (a) parallel runs appear side by side along the X-axis, and (b) sequential runs are arranged along the timeline in the Z-axis direction.

Jobs as Sub-blocks

Jobs within workflow runs are visualized as smaller rectangular blocks positioned on top of their respective workflow run. Each job is spatially arranged to preserve temporal relationships while avoiding visual overlap.

Track Assignment and Width Calculation:

The system uses adaptive track assignment to optimize horizontal space usage within each run:

1. Jobs are sorted chronologically by start time

2. Each job is assigned to the first available track without temporal overlap
3. Track width is dynamically calculated based on the total number of tracks:

$$jobWidth = \frac{availableWidth - (trackCount - 1) \times trackGap}{trackCount}$$

Formula Components:

- **jobWidth:** Width allocated to each individual job block
- **availableWidth:** Usable horizontal space within the run (runWidth - 1.0 units for margins)
- **trackCount:** Total number of parallel tracks needed for all jobs in the run
- **trackGap:** Fixed spacing of 0.5 units between adjacent job tracks

Job Positioning System:

Jobs are positioned using both temporal and spatial coordinates:

- **Temporal Positioning (Z-axis):** Job position along the timeline based on execution timestamps:

$$jobZPosition = jobZStart + \frac{jobLength}{2}$$

Formula Components:

- **jobZPosition:** Final Z-coordinate position of the job along the timeline
- **jobZStart:** Starting Z-position derived from the job's `started_at` timestamp
- **jobLength:** Temporal duration of the job converted to spatial units (jobZEnd - jobZStart)
- **Spatial Positioning (X-axis):** Jobs are distributed across tracks within the run's horizontal space:

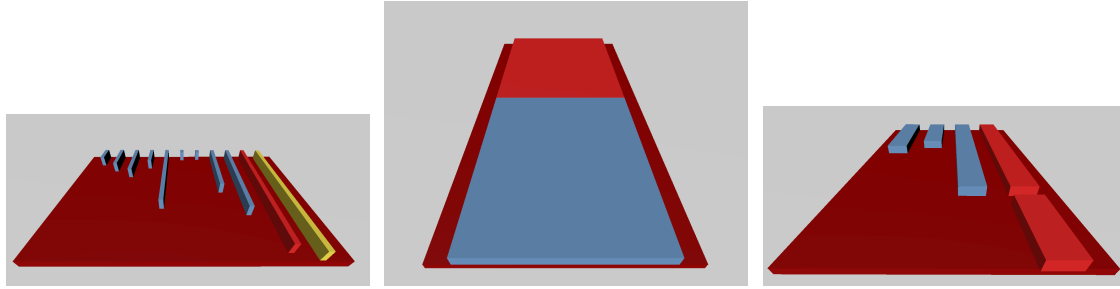
$$jobXPosition = runXPosition - \frac{runWidth}{2} + relativeXPosition + \frac{jobWidth}{2}$$

Formula Components:

- **jobXPosition:** Final X-coordinate position of the job within the 3D scene
- **runXPosition:** X-coordinate of the parent workflow run block
- **runWidth:** Total horizontal width of the parent run block
- **relativeXPosition:** Job's position within the run, calculated from track assignment
- **jobWidth:** Width allocated to each job based on track count and available space

Jobs are scaled proportionally to their temporal duration and positioned with a 0.5-unit horizontal margin within the run. Job materials indicate execution status through color coding, consistent with the workflow run visualization approach detailed in Section 4.6.

Figure 4.10 demonstrates job layout patterns within workflow runs, showing how the track assignment algorithm handles different execution scenarios.



(a) Parallel jobs distributed across multiple tracks within a single run. (b) Sequential jobs arranged chronologically along the timeline within a run. (c) Combined workflow run showing both parallel and sequential job execution patterns.

Figure 4.10: Job positioning within workflow runs: (a) parallel execution patterns use horizontal tracks, (b) sequential patterns follow temporal progression along the Z-axis, and (c) realistic workflows combining both parallel and sequential job dependencies.

Time Compression and Inactivity Management

The system compresses gaps longer than a specified threshold to avoid unnecessary spatial elongation during long periods of inactivity (default: 2 hours). The time compression threshold is configurable through the Unity Inspector interface, allowing for adjustment of the compression sensitivity based on repository activity patterns and user preferences. Gap detection aggregates all runs across all workflows into a single chronological list and identifies inter-run gaps exceeding the threshold.

For a time period to qualify for compression, it must represent complete inactivity across the entire repository, meaning no workflow runs exist during that period in any workflow.

When such a gap is detected, it is collapsed using a time portal. Events following the gap are shifted forward in Z-space by subtracting the cumulative duration of all prior compressed segments.

This repository-wide compression strategy ensures that the temporal-spatial mapping remains consistent and intuitive while effectively reducing navigation distances during periods of complete development inactivity.

Portal Implementation:

Each time-compression portal is instantiated as a semi-transparent blue vertical block. Portal thickness is scaled logarithmically based on the duration of the compressed gap:

$$thickness = baseThickness \times \log_{10} \left(\frac{gapHours}{minGapHours} \right)$$

Formula Components:

- **thickness**: Final thickness of the portal block in world units
- **baseThickness**: Minimum portal thickness for the smallest compressible gap
- **gapHours**: Duration of the compressed time gap in hours
- **minGapHours**: Minimum gap duration threshold (default: 2 hours)

The logarithmic scaling ensures that portals grow visually distinguishable but remain manageable in size, a 2-hour gap and a 200-hour gap don't result in proportionally massive visual differences that would disrupt the VR environment's scale and navigation.

Portals are positioned at the temporal boundaries of compressed gaps and extend across all workflow floors to maintain visual consistency throughout the factory environment.

Each portal contains two types of labels:

- A floating top label positioned above the portal for overhead viewing
- A world-anchored side label positioned at the left side of the factory for lateral viewing

Both labels display identical information including the duration of the compressed gap (e.g., "Time Jump: 4h 30min") and the start and end timestamps. The dual positioning ensures users can access temporal information from different viewing angles and navigation positions within the VR environment.

Figure 4.11 illustrates both label types positioned on time compression portals within the factory environment.



(a) Portal with floating top label showing time jump duration.



(b) Portal with world-anchored side label displaying start and end timestamps.

Figure 4.11: Time compression portal labeling system: (a) floating top labels indicate the duration of compressed time gaps, and (b) side labels provide precise timestamp information for the compressed interval boundaries.

These elements visually signal discontinuities in the timeline while preserving user awareness of the skipped interval. The system significantly reduces required travel distance in VR by compressing inactive periods, allowing users to reach areas of interest more quickly while maintaining rendering efficiency.

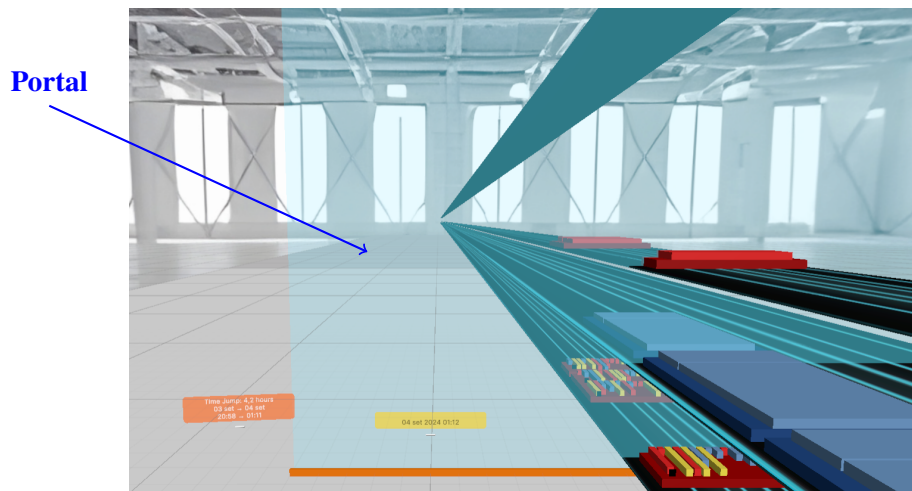


Figure 4.12: Time compression portal spanning the factory environment, showing the semi-transparent blue structure that signals temporal discontinuities in the workflow timeline.

Scene Hierarchy and Prefab Management

To maintain performance and modularity, all elements are instantiated from prefabs and organized hierarchically:

- `Workflows/` contains all workflow floors
- Each floor contains its `ConveyorBelt/` and associated runs
- Each run contains its `Jobs/` sub-objects
- Labels, materials, and panels are instantiated as children to ensure positional coherence

4.4 Audio Layer

The audio layer of the system provides spatial auditory cues to assist users in detecting failed workflow runs and navigating the immersive environment more efficiently. This section outlines the implementation of 3D audio features, covering spatial configuration, triggering logic, lifecycle management, and accessibility considerations.

Spatial Audio for Failure Detection

Failure-specific sounds augment visual indicators and improve user awareness in complex workflow scenes. For each run with a conclusion marked as `"failure"`, a 3D audio source is instantiated at runtime and positioned at the corresponding visual marker location in the scene. These sources emit a continuous siren-like alert, emulating industrial warning sounds.

Figure 4.13 demonstrates the audio system in action, showing a sound icon positioned above a failed workflow run while successful runs remain silent.

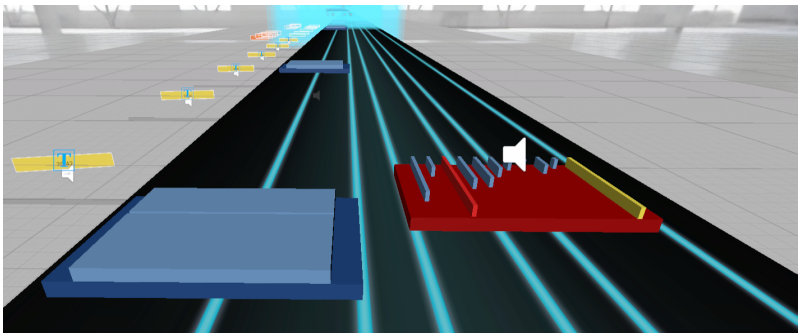


Figure 4.13: Spatial audio visualization showing a sound icon above a failed workflow run while successful runs have no audio indicators, demonstrating selective audio feedback for failure detection.

The audio source is fully spatialized using `spatialBlend = 1.0`, which ensures positional rendering in the user’s headset. Distance-based attenuation follows Unity’s logarithmic roll-off model: full volume from 0m to 1m, falls off logarithmically to silence between 1m and 10m, and is silent when more than 10m away. This configuration avoids distracting users with audio from distant workflow floors while preserving immersion. Table 4.4 summarizes the key properties used in the spatial audio setup.

Table 4.4: Spatial Audio Configuration for Failure Events

Parameter	Value / Description
<code>spatialBlend</code>	1.0 (Fully 3D)
<code>minDistance</code>	1.0 (Full volume within 1 meter)
<code>maxDistance</code>	10.0 (Silent beyond 10 meters)
<code>rolloffMode</code>	Logarithmic
<code>loop</code>	Enabled (continuous playback)
<code>dopplerLevel</code>	1.0 (enabled, minimal effect)
<code>volume</code>	0.7 (moderate intensity)

Sound Triggering and Lifecycle

Failure sounds are instantiated programmatically during the construction of the visualization scene. The system prevents redundant audio sources by maintaining an internal registry of already-triggered runs. Each audio source loops continuously until the associated run object is destroyed or the system is reset. This ensures that sound output remains consistent with the current visual state without leaving orphaned audio objects in memory.

The `AudioManager` class serves as a centralized controller, managing the loading of sound assets (from `Resources/Sounds/run_failure`), creation of audio `GameObjects`, and cleanup of sources when runs are removed, or scenes reloaded. Table 4.5 outlines the typical lifecycle states of an audio source in the system.

Table 4.5: Failure Audio Source Lifecycle Events

Event	Condition	System Behavior
Trigger	Run.conclusion == "failure"	Instantiate audio source at run position
Duplicate Check	Run ID already active	Prevent second audio instance
Cleanup	Run destroyed or scene reset	Destroy audio GameObject
Mute Toggle	User toggles sound off	Mute all active audio sources
Unmute Toggle	User toggles sound on	Restore volume on all sources

UI Interaction Sounds

User interface interactions are supported by default click sounds provided by the XR Interaction Toolkit. These are triggered when users interact with UI elements like toggles and dropdowns. While these sounds were not customized, they contribute to the responsiveness of the interface and reinforce action confirmation. No additional configuration was applied to these assets beyond what the toolkit offers.

Audio Accessibility and User Control

A global toggle allows users to turn all auditory feedback on or off. The toggle is accessible through the floating menu panel attached to the user's non-dominant hand and takes immediate effect. When audio is disabled, all existing and future failure sounds are muted programmatically by setting each source's `mute` property. This ensures the user's preference is consistently respected across all audio contexts.

For accessibility and testing, the same mute functionality is also mapped to a keyboard key (M) in non-VR builds. The toggle state is applied retroactively to active sounds and is preserved throughout the session.

Integration and Performance Considerations

Audio feedback is tightly integrated into the visualization layer, with each failure sound bound to a corresponding 3D run object. Positioning is matched precisely to the run's conveyor belt location, ensuring coherence between visual and auditory cues. The system avoids unnecessary performance overhead by limiting sound activation to failed runs and ensuring the timely destruction of inactive sources.

This audio layer enhances situational awareness by converting workflow failure events into spatially localized sounds, reducing the need for constant visual scanning and supporting a more immersive and efficient CI/CD environment exploration.

4.5 Interaction Layer

The interaction layer enables users to navigate and explore the immersive CI/CD visualization using multiple locomotion techniques and context-sensitive inspection tools. Designed to accommodate user preferences and accessibility needs, the system integrates seamlessly with Unity's XR Interaction Toolkit to provide a VR interaction experience.

Navigation Systems

- **Physical Movement:** Room-scale tracking through Unity's `XROrigin` component enables natural movement within the user's physical play space, with automatic head and controller tracking
- **Joystick Locomotion:** Smooth movement via the right-hand controller joystick, with configurable rotation (snap or smooth) mapped to the secondary axis for continuous navigation
- **Teleportation:** Point-and-click teleportation maintains correct orientation and world-space alignment, providing a comfort-focused alternative to continuous motion
- **Floor Navigation:** Vertical movement between workflow floors is supported through two mechanisms:
 - **Button Controls:** X and Y buttons on the left-hand controller move down or up one floor respectively, with each transition spanning exactly 3.5 world units to match the factory layout spacing
 - **Dropdown Menu:** A dynamically populated UI dropdown lists all workflows using the format "`FloorIndex` - `WorkflowName`", enabling immediate teleportation to any floor

Figure 4.14 illustrates the controller button mapping for floor navigation and joystick-based locomotion.



Figure 4.14: VR controller input mapping: (a) left controller with X (down arrow) and Y (up arrow) buttons for floor navigation with directional indicators, and (b) right controller with joystick for horizontal movement and rotation within the factory environment.

User Interface Menu

A floating UI menu is attached to the user's non-dominant hand, providing centralized access to system controls and navigation options. The menu remains consistently accessible throughout the VR experience while maintaining an unobtrusive profile during exploration.

Menu Components: The hand-mounted menu includes two primary controls:

- **Audio Toggle:** Enables users to turn spatial audio feedback on or off for workflow failure events, with immediate effect on all active and future audio sources
- **Floor Selection Dropdown:** A dynamically populated dropdown listing all workflows using the format "FloorIndex - WorkflowName", enabling instant teleportation to any workflow floor while maintaining the same XZ position

The dropdown menu is automatically updated when new workflow data is loaded, ensuring that floor navigation options remain synchronized with the current visualization state. Floor selection through the dropdown provides an alternative to the button-based elevator system, particularly useful for quick navigation across multiple workflow levels.

Figure 4.15 shows the hand-mounted menu interface positioned on the user's left hand (non-dominant hand in right-handed mode).

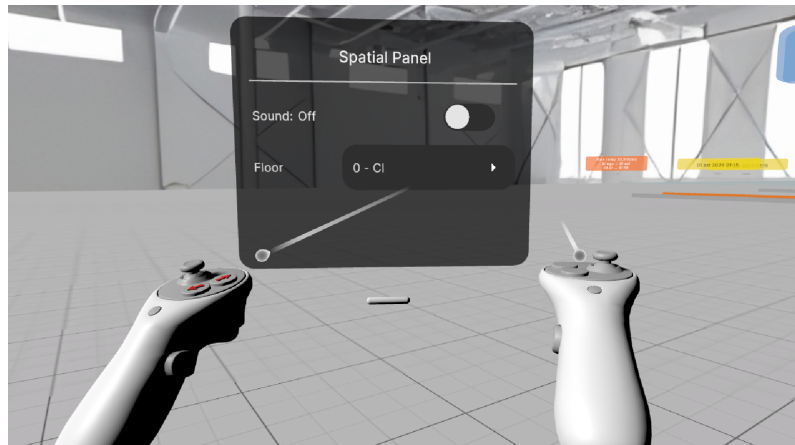


Figure 4.15: Hand-mounted user interface menu attached to the left hand, displaying audio controls and floor selection dropdown for convenient access during VR exploration.

The menu's hand-mounted design ensures that controls remain within easy reach without cluttering the main visualization space, supporting interaction while maintaining focus on the CI/CD data exploration.

Data Inspection and Information Panels

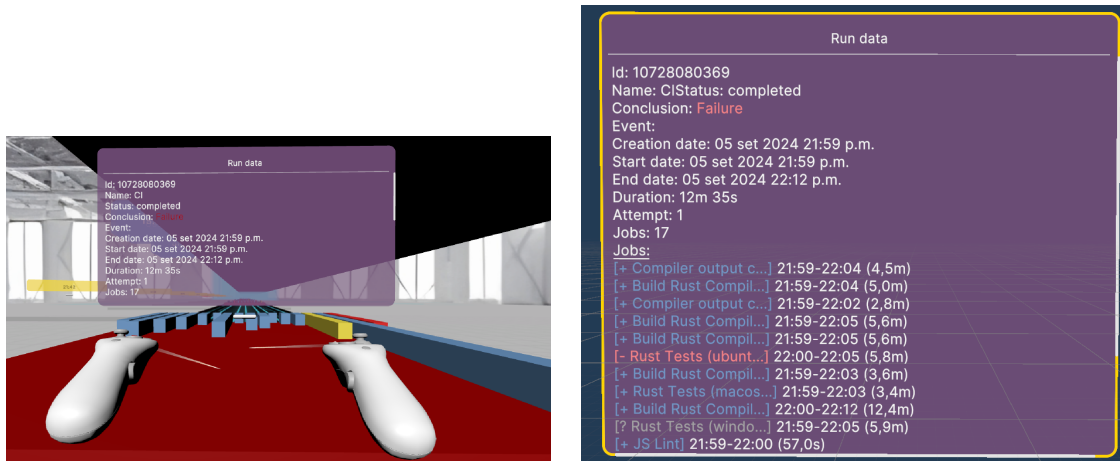
The system provides data inspection tools to examine workflow details directly within the VR environment:

Hover-Based Information Panels: When users hover over workflow run blocks with the controller pointing ray, detailed information panels showing detailed information about the workflow run. These panels are automatically managed through the XR Interaction Toolkit and disappear when hover interaction ends, ensuring clean visual transitions during exploration.

Panel Content and Layout: Each information panel displays workflow run metadata organized into two main sections:

- **Run Overview:** ID, name, status, conclusion (with color coding in normal mode), trigger event, creation/start/end dates, duration, attempt number, and total job count
- **Job Breakdown:** Individual job details including name (truncated for display), color-coded execution status, and compact execution time formatting

Figure 4.16 demonstrates the information panel system in both in-scene context and complete content view.

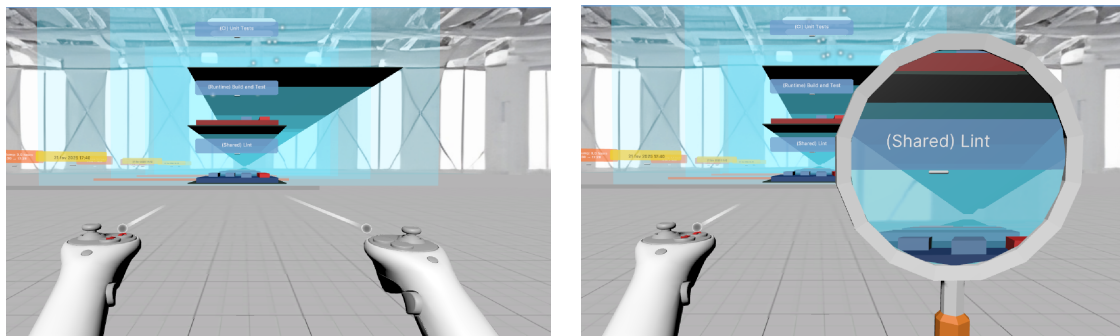


(a) Information panel positioned in-scene above a work-flow run for VR legibility. (b) Complete panel content showing all available run and job metadata.

Figure 4.16: Information panel system: (a) spatial positioning within the VR environment and (b) metadata display with job-level breakdowns.

Magnifier Tool: A virtual magnifier attached to the user’s dominant hand provides enhanced accessibility for detailed inspection. The tool toggles visibility via button input and temporarily hides the controller model during use to maintain focus on magnified content.

Figure 4.17 demonstrates the magnifier tool’s effect by comparing the view without and with magnification when inspecting detailed elements within the VR environment.



(a) View without magnifier showing standard text readability. (b) Magnifier tool in use, enhancing text clarity and detail visibility.

Figure 4.17: Comparison of text inspection with and without the magnifier tool, demonstrating enhanced accessibility for detailed content examination in the VR environment.

4.6 Visual Encoding and Accessibility

The visual encoding system of the prototype uses color and material differentiation to communicate status information clearly while supporting accessibility for users with color vision disabilities. This section describes the implemented color logic, accessibility options, and material

management strategies used to ensure consistency and usability in the immersive environment.

Status-Based Color Encoding

Color is the primary channel for encoding the status of workflow runs and individual jobs. The color palette selection process prioritized accessibility and visual hierarchy considerations before determining the final color assignments.

Palette Selection and Validation

The color palette is derived from the Tango color scheme but was modified to avoid red-green contrasts, which are problematic for users with the most common forms of color blindness. The selection process focused on blue, yellow, and red as the base hues to ensure intuitive status mapping while maintaining accessibility. The final palette was validated using color vision simulation tools, including the BYU Colorblind Image Tester⁵ and the David Math Logic colorblind simulator⁶. These tests confirmed that status indicators remained distinguishable under deuteranopia, protanopia, and tritanopia simulations.

Figure 4.18 shows the validation process using the BYU Colorblind Image Tester. The original palette was submitted for evaluation, and the simulation result for 80% deuteranopia showed the tool's prediction of "friendly" with 100% confidence, confirming full color distinguishability for users with red-green vision deficiency.

⁵https://bioapps.byu.edu/colorblind_image_tester

⁶<https://davidmathlogic.com/colorblind>

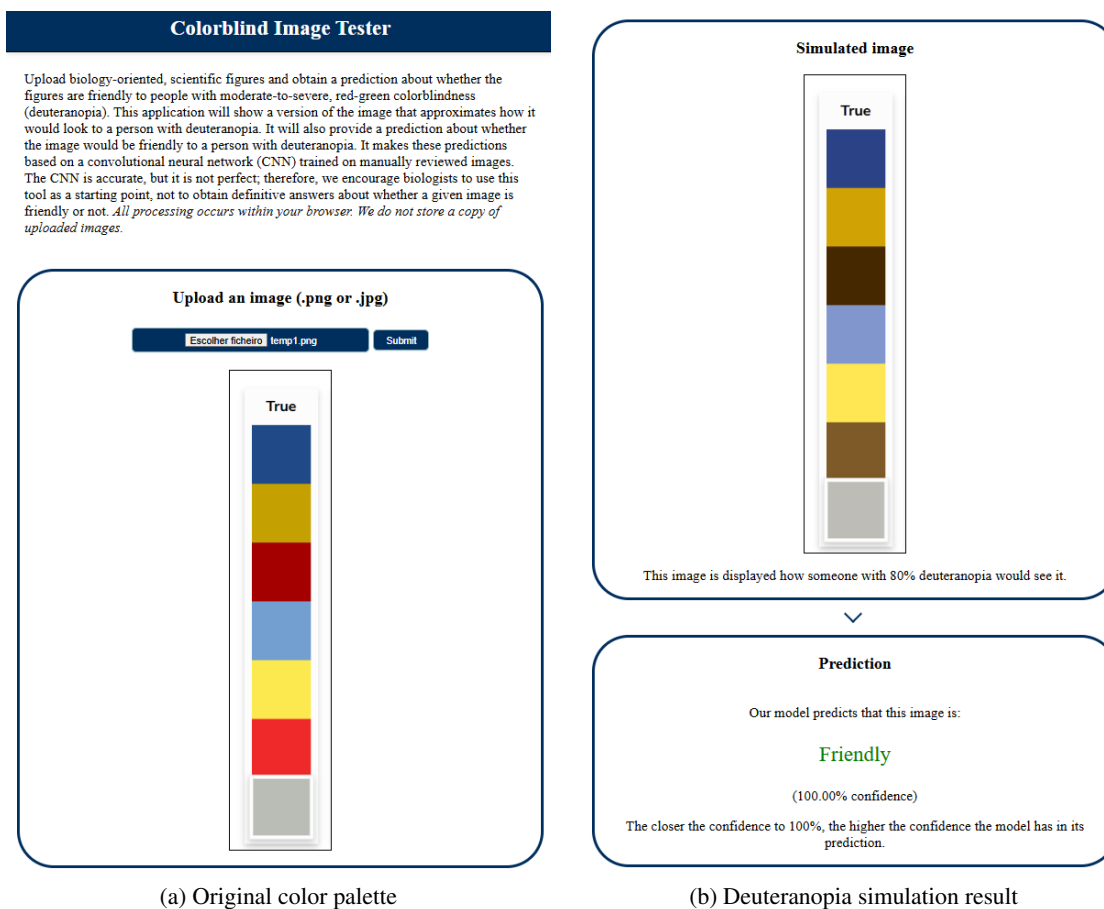


Figure 4.18: Color accessibility validation using the BYU Colorblind Image Tester showing original palette and deuteranopia simulation results.

Based on this validation process, a color scheme was established and applied consistently across visualization layers. Workflow-level blocks adopt darker tones, while job sub-blocks use lighter shades for contrast and visual hierarchy. Table 4.6 summarizes the final color assignments used in the prototype.

Table 4.6: Status-Based Color Encoding Used in the Prototype

Status	Workflow Runs (Hex)	Jobs (Hex)
Success	#204A87 (dark blue)	#729FCF (light blue)
In Progress	#C4A000 (mustard yellow)	#FCE94F (light yellow)
Failure	#A40000 (dark red)	#EF2929 (bright red)

These colors are applied dynamically to each visual element during the data parsing stage of scene construction. Job-level colors are adjusted slightly to distinguish them from their parent blocks while maintaining consistent semantics.

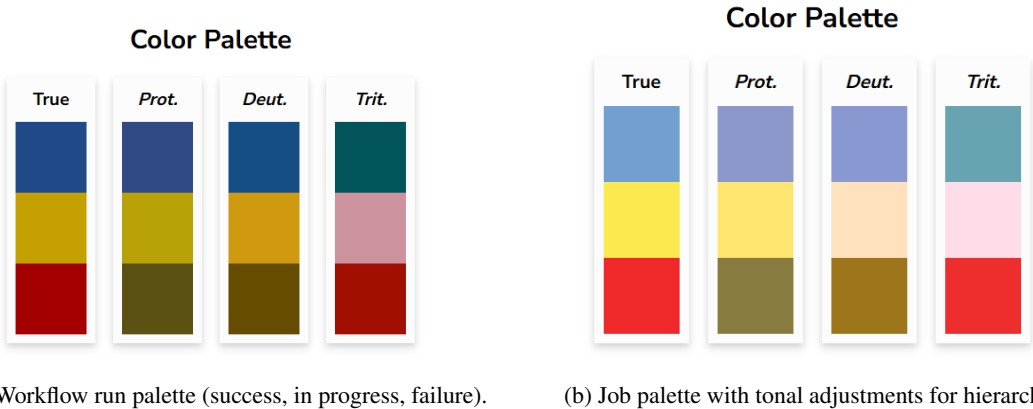


Figure 4.19: Comparison of the outcome-based color schemes used for workflow runs and job elements. Both palettes maintain consistent semantics and are adjusted for perceptual separation.

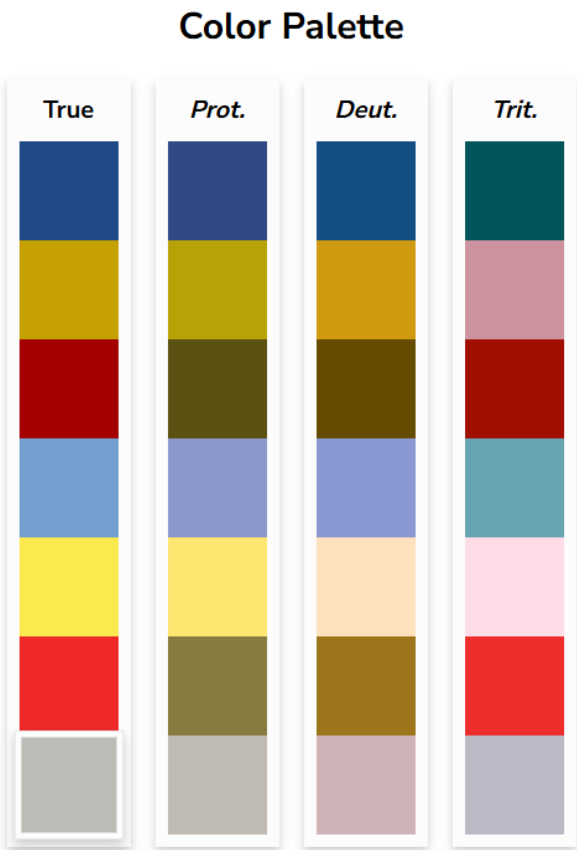


Figure 4.20: Final palette used in the prototype, combining run-level, job-level, and neutral UI colors.

4.7 Performance Considerations

Some performance strategies were implemented to ensure stable operation on VR hardware. This section outlines the main optimizations used in the visualization system.

Material Management and Reuse

The system loads materials once during initialization and reuses them across all workflow runs and jobs with identical status. Materials are loaded from Unity's Resources system during the Awake method and cached for reuse, reducing memory overhead by sharing material references rather than creating duplicate instances for each visual element.

Hierarchical Scene Organization

All scene objects are structured hierarchically under workflow-level parent containers. This organization enables efficient batch operations when switching between datasets or clearing the scene, as entire workflow hierarchies can be destroyed or deactivated as single units rather than requiring individual object management.

Conditional Audio Processing

Audio sources for failed workflow runs are only instantiated when failures are detected, avoiding unnecessary audio processing overhead. The audio system includes duplicate prevention to ensure only one audio source exists per failed run, preventing audio overlap and reducing processing load.

Simplified Timeline Rendering

Timeline markers are created as basic primitive cubes rather than complex mesh objects, reducing rendering complexity. Marker creation is limited to 30-minute intervals, avoiding excessive draw calls for dense temporal data.

Resource Loading Strategy

Visual assets including materials and prefabs are loaded from Unity's Resources system during initialization rather than per-object instantiation. This front-loading approach prevents frame rate drops during scene construction, though it requires all assets to be available at startup.

During data loading operations, users remain in the VR environment and can navigate freely, but the workflow visualization elements are not displayed until the complete dataset has been processed and the scene construction is finished. This approach ensures visual consistency and prevents partial or incomplete representations during the loading phase.

These optimizations focus on reducing redundant operations and maintaining clean object hierarchies rather than complex performance systems. The implementation prioritizes simplicity and reliability over advanced optimization techniques.

4.8 Summary

This chapter documents the technical implementation of the VR-based CI/CD visualization prototype. The system implements a four-layer architecture with data handling (GitHub API and JSON processing), visualization (3D factory metaphor with workflows as floors, runs as blocks, jobs as sub-blocks), interaction (VR navigation, hover panels, accessibility tools), and audio (spatial failure alerts). Key features include temporal-to-spatial mapping, lane assignment algorithms, time compression portals, colorblind-validated visual encoding, and performance optimizations for VR hardware.

The following chapter evaluates the system's performance, usability, and effectiveness in supporting CI/CD pipeline analysis compared to traditional visualization approaches.

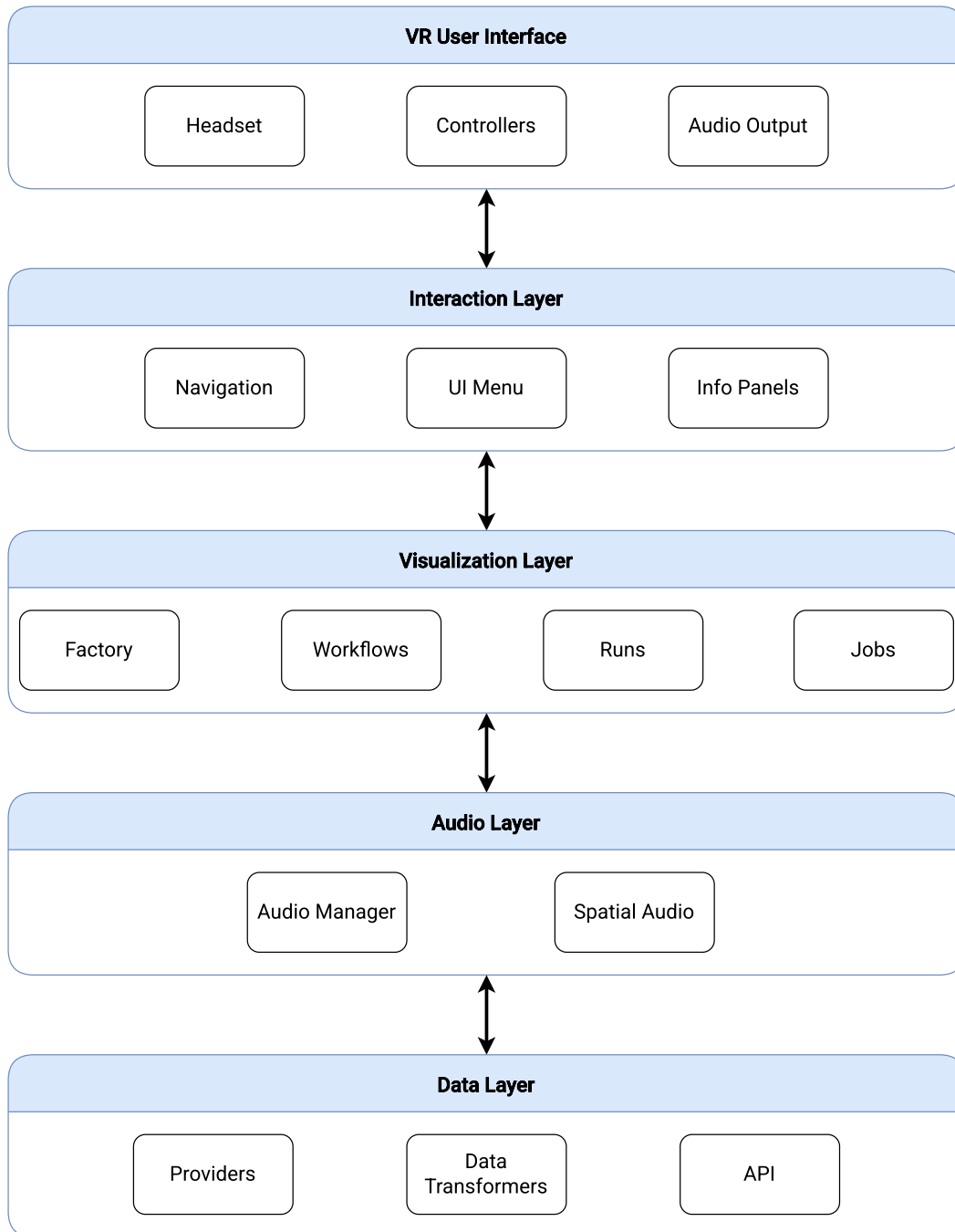


Figure 4.1: High-level architecture of the prototype system, showing the main subsystems and their relationships.

Chapter 5

Evaluation

A comparative user study was conducted to evaluate the effectiveness and usability of the VR-based CI/CD visualization system against GitHub Actions' traditional web interface. The study aimed to answer three key research questions:

- **RQ1:** Does a VR-based visualization system help users more easily identify and interpret CI/CD workflow structure and failures compared to a traditional 2D interface?
- **RQ2:** Does using spatial layout in a 3D environment contribute to reduced task time and fewer errors when exploring workflow execution data?
- **RQ3:** How do users perceive the ease of use, learnability, and overall satisfaction with the VR-based visualization system compared to traditional tools?

5.1 Evaluation Setup

The evaluation took place in controlled laboratory environments at the Faculty of Engineering, University of Porto (FEUP), specifically in the GIG laboratory (room I220) and university amphitheaters (B011 and B031). Twenty participants with varying levels of CI/CD and VR experience completed standardized tasks in both environments, providing quantitative performance data and qualitative feedback to address the research questions.

Hardware and Technical Configuration

The VR system was deployed using a Meta Quest 3 headset operating in PC VR mode, connected to a laptop computer running the Unity-based prototype. The test machine was equipped with an AMD Ryzen 7 5800H processor (8 cores, 16 threads, base speed 3.20 GHz), 16 GB of RAM, and an NVIDIA GeForce RTX 3060 Laptop GPU with 6 GB of dedicated memory.

For the comparative condition, participants used GitHub Actions through the Google Chrome browser with the Dark Tritanopia theme enabled. Both conditions utilized identical underlying

data from the Facebook/relay repository¹. The workflow runs that were considered were those that occurred between August 28, 2024, to September 1, 2024. This date range was selected after analyzing the repository's commit frequency to ensure sufficient workflow activity and consistency across all test sessions.

Dataset and Temporal Constraints

The system used a fixed temporal dataset rather than live GitHub data to maintain consistent test conditions across all evaluation sessions. The VR prototype filtered workflow runs to the specified date range programmatically, while GitHub Actions access was provided through a filtered URL² targeting the specified date range (alternatively, this filtering could be applied directly through GitHub's interface date filters). This approach prevented variations in available data between testing sessions while providing realistic CI/CD activity patterns from an active open-source repository.

Environment and Physical Setup

All evaluation sessions were conducted seated to accommodate laboratory constraints and ensure participant comfort during the 40-45 minute testing periods. The VR setup provided adequate room-scale tracking within the seated boundaries, while the desktop configuration used standard keyboard and mouse interaction with a desktop monitor. Figure 5.1 shows the typical evaluation configuration with a participant using the VR system. The Meta Quest 3 headset was connected to the laptop computer, allowing the researcher to monitor the participant's VR experience on the external display for observation and assistance purposes.

5.2 Participants

The evaluation was conducted with 20 participants recruited from the Faculty of Engineering, University of Porto (FEUP). The target population consisted primarily of individuals with academic or professional backgrounds in computer science and related technical fields, ensuring familiarity with software development concepts while maintaining varied levels of experience with CI/CD systems and virtual reality technology.

Recruitment and Demographics

Participants consisted mainly of undergraduate and graduate students in computer engineering programs. The sample included 18 participants from computer engineering backgrounds and 2 participants from biomedical engineering.

The demographic composition was as follows: 14 male participants (70%) and 6 female participants (30%), with ages distributed predominantly in the 20-29 age range (19 participants, 95%)

¹<https://github.com/facebook/relay/actions>

²<https://github.com/facebook/relay/actions?query=created:2024-08-28..2024-09-01>



(a) Participant with VR headset and controllers (b) VR system display visible to researcher

Figure 5.1: Evaluation setup showing participant interaction with the VR system. (a) Front view of participant wearing Meta Quest 3 headset with hand controllers. (b) Researcher's view of the VR environment displayed on the laptop screen for monitoring and assistance.

and one participant in the 30-39 age range. Regarding educational background, 17 participants held bachelor's degrees (85%), 2 held master's degrees (10%), and 1 held a doctorate (5%).

Handedness distribution showed 18 right-handed participants (90%) and 2 left-handed participants (10%), with no reported ambidextrous participants. Importantly, no participants reported color vision deficiencies.

Experience Profiles

The participant pool demonstrated varied experience levels across the key technology domains relevant to the study:

Virtual Reality Experience: The majority of participants had limited VR exposure, with 4 participants (20%) having never used VR, 13 participants (65%) having used it a few times (less than 5 times), 1 participant (5%) reporting monthly use, 1 participant (5%) reporting frequent use, and 1 participant (5%) reporting weekly use. This distribution provided a realistic representation of general VR adoption levels among technical users.

CI/CD Pipeline Experience: Experience with CI/CD concepts varied significantly, reflecting real-world knowledge distribution in academic and early-career settings. 6 participants (30%) did not know what CI/CD pipelines were, 4 participants (20%) had never used them, 6 participants

(30%) had limited experience (less than 5 times), 1 participant (5%) reported monthly use, 2 participants (10%) reported frequent weekly use, and 1 participant (5%) reported daily use.

GitHub Actions Experience: GitHub Actions familiarity was similarly distributed, with 7 participants (35%) not knowing what GitHub Actions was, 6 participants (30%) having never used it, 4 participants (20%) having limited experience, 2 participants (10%) reporting monthly use, 1 participant (5%) reporting frequent use, and 0 participants reporting daily use.

Participant Selection Rationale

This experience distribution was intentionally diverse to evaluate system usability across different knowledge levels, from novice users requiring comprehensive tutorials to experienced users who could provide informed comparisons with existing tools. The inclusion of participants with limited CI/CD experience reflects realistic academic environments where students and researchers may encounter these systems without extensive prior exposure.

The predominantly young, technically-oriented sample aligns with the target user base for immersive development tools while providing sufficient experience variation to assess both learnability for novices and efficiency gains for experienced users.

5.3 Task Design

The evaluation tasks were designed to assess key aspects of CI/CD pipeline analysis that developers commonly encounter when working with workflow systems. Based on the literature review findings that developers struggle with failure detection, temporal pattern recognition, and navigating complex pipeline structures [36], four specific tasks were developed to test different cognitive and perceptual capabilities of both visualization approaches.

Task Selection Rationale

The tasks were designed to represent common CI/CD analysis activities identified in the state-of-the-art review:

- **Temporal Navigation:** Developers frequently need to locate recent executions to understand current pipeline status and identify when issues began occurring
- **Failure Detection:** Rapid identification of failed runs is critical for maintaining system reliability and reducing downtime
- **Job-Level Debugging:** Understanding which specific jobs caused failures is essential for targeted troubleshooting and efficient problem resolution
- **Multimodal Awareness:** Testing whether spatial audio can augment visual information for failure detection, addressing the identified gap in multimodal CI/CD interfaces

These task categories align with the visualization goals established in the conceptual design chapter (3): temporal clarity, failure highlighting, hierarchical representation, and multimodal feedback integration.

Task Descriptions and Objectives

Four tasks were administered to each participant, with the first three performed in both VR and GitHub Actions environments and the fourth exclusive to the VR system:

Task 1 - Temporal Workflow Navigation: "Find and tell me at what time was the most recent run of the workflow with the name 'Update Cargo.lock'."

This task evaluates users' ability to navigate temporal information and locate specific workflow instances. It tests the effectiveness of the spatial timeline representation in VR versus GitHub Actions' chronological listing approach. The task requires participants to understand workflow organization and temporal ordering.

Task 2 - Failure Detection and Identification: "Find the most recent failed run in the workflow with the name 'CI' and tell me the last three digits of its ID."

This task assesses rapid failure detection capabilities, a critical skill for maintaining CI/CD pipeline health. It evaluates how effectively each interface highlights failed executions and supports quick identification. The task tests both visual scanning efficiency and the cognitive load required to distinguish failure states from successful executions.

Task 3 - Job-Level Debugging: "Find the jobs that caused the run you found in Task 2 to fail and tell me their names."

This task examines users' ability to drill down from workflow-level failures to specific job-level causes, representing the debugging workflow developers use to isolate and fix pipeline issues. It tests the hierarchical navigation capabilities of each system and evaluates how well the interfaces support the transition from overview to detailed inspection.

Task 4 - Spatial Audio Effectiveness (VR Only): "Only using sound, show me an example of a run that is failing."

This task exclusively tests the spatial audio system implemented in the VR prototype by removing all visual status indicators. For this task, a specialized "no color mode" was activated where all workflow runs and jobs are rendered in a uniform gray color from the established palette, eliminating the normal success/failure color coding. Additionally, all textual status information was removed from the workflow run information panels, including conclusion indicators (success, failure, in-progress) and job-level status descriptions. This removal of visual and textual cues forces participants to rely entirely on the 3D spatial audio feedback for failure detection. The task addresses the research gap in multimodal CI/CD interfaces and evaluates whether spatial audio can serve as an effective standalone channel for failure awareness in immersive environments.

5.4 Methodology

The evaluation used a within-subjects comparative design where all participants experienced both the VR prototype and GitHub Actions interface. This approach allows for a direct comparison of user performance and preferences while controlling for individual differences in technical background and cognitive abilities.

Experimental Design

Participants were assigned to one of two testing sequences to minimize order effects and learning bias using alternating assignments based on participant ID. Participants with odd ID numbers (1, 3, 5, ..., 19) began with GitHub Actions tasks followed by VR tasks, while participants with even ID numbers (2, 4, 6, ..., 20) began with VR tasks followed by GitHub Actions tasks. This counterbalancing ensured equal exposure to both starting conditions across the participant pool and mitigated potential advantages from prior task familiarity.

Each participant completed all four tasks in both environments where applicable, with Task 4 being exclusive to the VR system due to its spatial audio requirements. The task order remained consistent within each environment to maintain comparability across participants.

Session Structure and Procedure

Each evaluation session followed a standardized protocol lasting 40-45 minutes outlined in Table 5.1:

Table 5.1: Evaluation Session Structure

Phase	Duration	Activities
Pre-Session	5 min	Consent, demographics, comfort assessment
First Environment	15-20 min	CI/CD concepts tutorial (3 min) Environment-specific tutorial & sandbox (5-8 min) Tasks 1-3 execution & timing Post-environment questionnaire
Transition	2-3 min	Environment switch, technical setup
Second Environment	15-20 min	Environment-specific tutorial & sandbox (5-8 min) Tasks 1-3 execution & timing Task 4 (VR only) Post-environment questionnaire
Post-Session	3-5 min	Comparative discussion, final feedback

Tutorial Content and Structure

All participants completed a standardized three-part tutorial sequence to ensure consistent baseline knowledge:

CI/CD Conceptual Foundation: Participants viewed a tutorial video³ explaining CI/CD principles and GitHub Actions architecture, covering:

1. Continuous Integration and Continuous Delivery concepts and benefits
2. Workflow definition as automated task scripts triggered by repository events
3. Workflow run creation and execution lifecycle
4. Job structure as grouped steps running on virtual machines
5. Status hierarchies (success, failure, in-progress) at both run and job levels
6. Sequential versus parallel execution patterns

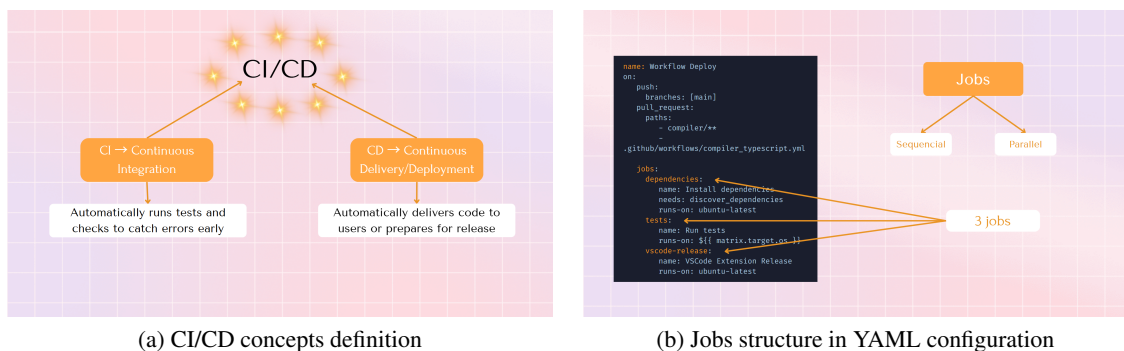


Figure 5.2: Screenshots from the CI/CD tutorial video showing fundamental concepts and workflow structure.

GitHub Actions Tutorial: Participants viewed a comprehensive video tutorial⁴ covering:

1. Navigation through the Actions tab and workflow interface
2. Understanding workflow lists and run displays
3. Status interpretation using color and symbol indicators
4. Filtering capabilities and metadata examination
5. Job-level exploration and failure diagnosis techniques

VR System Tutorial: The VR tutorial began with a video introduction⁵ to the factory metaphor, covering:

³CI/CD Tutorial: https://drive.google.com/file/d/1w24Aqst4MKraZrxWTt0CiWDUR-Ivp4hn/view?usp=drive_link

⁴GitHub Actions Tutorial: https://drive.google.com/file/d/1QDtmOg1TihhL4s5tAWtXwgt5tW2ZXGov/view?usp=drive_link

⁵VR System Tutorial: https://drive.google.com/file/d/1lDUxcYPoX6NkObPheZaNpwDqKthZrtd5/view?usp=drive_link

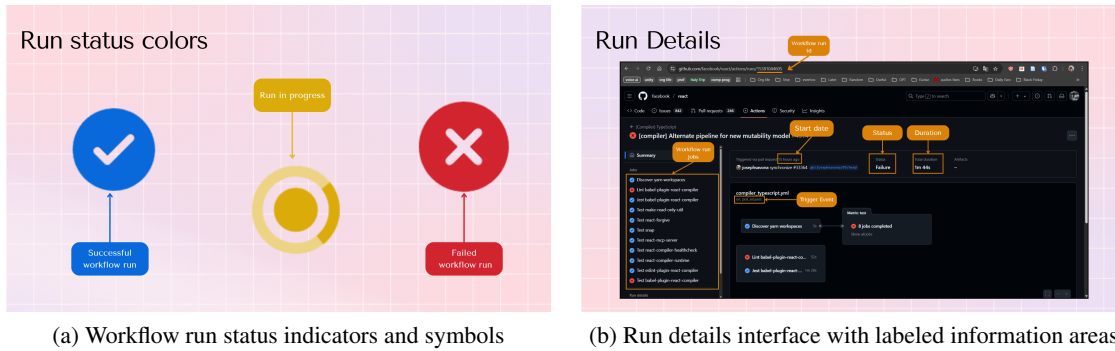


Figure 5.3: Screenshots from the GitHub Actions tutorial video demonstrating interface navigation and status interpretation.

1. Spatial layout principles (workflows as floors, time as depth)
2. Visual encoding (color coding, timeline markers)
3. Time compression portals and navigation aids
4. Interaction techniques and audio feedback systems

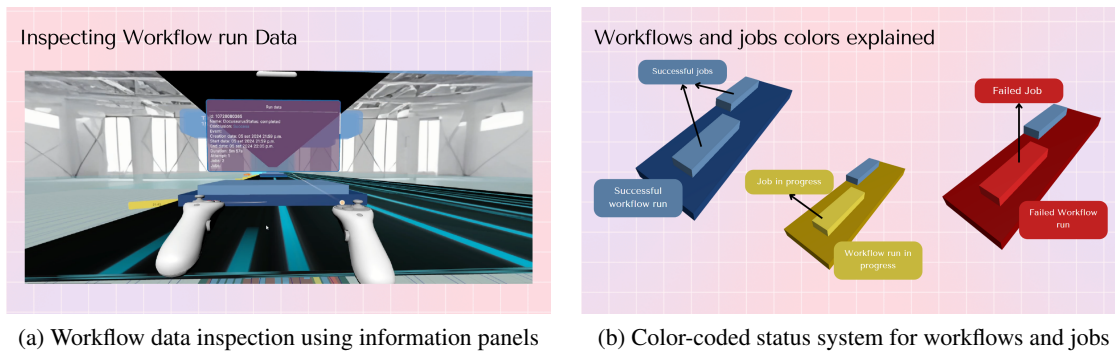


Figure 5.4: Screenshots from the VR system tutorial video showing interaction techniques and visual encoding principles.

Tutorial and Sandbox Environments

Tutorial delivery was adapted to participant experience levels. While all participants were initially scheduled to view the CI/CD conceptual foundation, 2 participants with professional CI/CD experience requested to bypass both the foundational CI/CD tutorial and GitHub Actions tutorial. These participants were permitted to proceed directly to the sandbox exploration phase to avoid redundant instruction and optimize session efficiency. After completing their respective tutorial sequences, all participants engaged in guided sandbox exploration using different datasets to optimize learning while preserving evaluation integrity:

GitHub Actions: Sandbox practice occurred within the Facebook/react repository, providing a realistic workflow environment distinct from evaluation tasks.

VR System: Sandbox exploration utilized a custom synthetic dataset inspired by the Facebook/react repository, designed to demonstrate all system capabilities, including time compression portals, diverse workflow states, and spatial audio feedback.

Data Collection

Data collection captured both performance and subjective measures as outlined in Table 5.2. Performance metrics included task completion times (measured from instruction delivery to answer provision), binary success/failure verification, and navigation pattern observations. Subjective assessment used structured post-environment questionnaires including the SUS [27] questionnaire translated to Portuguese [30] and VR-specific spatial audio evaluation questions on 5-point Likert scales [28]. Open-ended response fields captured additional qualitative insights.

Table 5.2: Data Collection Framework

Data Type	Measure	Collection Method
Performance	Task completion time	Stopwatch measurement (task start to answer)
	Task accuracy	Binary success/failure verification
	Navigation patterns	Real-time researcher observation
Subjective	System usability	SUS questionnaire (Portuguese) [30]
	Spatial audio effectiveness	5-point Likert scales (VR only)
	Qualitative insights	Open-ended response fields

Standardization and Controls

Several measures ensured consistent testing conditions:

- **Identical Dataset:** Both environments accessed the same Facebook/relay repository data filtered to August 28 - September 1, 2024
- **Standardized Instructions:** Task descriptions read verbatim to each participant with clarification allowed only for general procedural questions
- **Technical Consistency:** VR headset calibration performed for each participant; browser zoom and display settings standardized for GitHub Actions
- **Documentation Protocol:** Real-time note-taking by the researcher with post-session verification of recorded timings and responses

Ethical Considerations and Participant Comfort

The study protocol prioritized participant safety and comfort:

- **VR Safety:** Regular comfort checks during VR sessions with immediate discontinuation available upon request
- **Data Privacy:** All participant responses anonymized with secure storage of collected data
- **Voluntary Participation:** Participants informed of their right to withdraw at any time without consequence

This methodology provides a comprehensive framework for comparing the two visualization approaches while maintaining scientific rigor and participant welfare throughout the evaluation process.

5.5 Results

This section presents the findings from the comparative evaluation of the VR-based CI/CD visualization system against GitHub Actions' native interface. The analysis encompasses both quantitative performance metrics and qualitative user feedback collected from 20 participants across four evaluation tasks.

Prior to conducting comparative analyses, the collected performance data underwent normality testing to determine appropriate statistical methods. The Shapiro-Wilk [40] test was applied to task completion times for each environment, as this test is particularly suitable for small sample sizes ($n = 20$) and provides a reliable assessment of distributional assumptions.

The normality analysis revealed mixed distributional patterns across tasks. Task 1 completion times demonstrated significant deviations from normality in both environments ($p = 0.043$ for GitHub Actions, $p = 0.005$ for VR), indicating positively skewed distributions with some participants requiring substantially longer completion times. In contrast, Tasks 2 and 3 showed normal distributions across both environments (all $p > 0.05$), while Task 4 (VR-only) also followed a normal distribution ($p = 0.247$).

These findings determined the statistical approach for subsequent analyses: non-parametric Wilcoxon [47] signed-rank tests were employed for Task 1 comparisons due to non-normal distributions, while parametric paired t-tests were used for Tasks 2 and 3. McNemar's tests [31] were utilized to assess differences in binary outcomes between paired samples for success comparisons. Task 4, being exclusive to the VR environment, was analyzed using descriptive statistics and correlation analysis.

5.5.1 Task Performance Analysis

Task performance was evaluated across two primary dimensions: completion times and success rates. The following subsections present detailed analysis of task completion times, success rate comparisons, and an overall performance summary.

Task Completion Times

Task completion time analysis revealed significant performance differences between the VR and GitHub Actions interfaces, with patterns varying across different task types. Figure 5.5 presents the completion time distributions for all tasks.

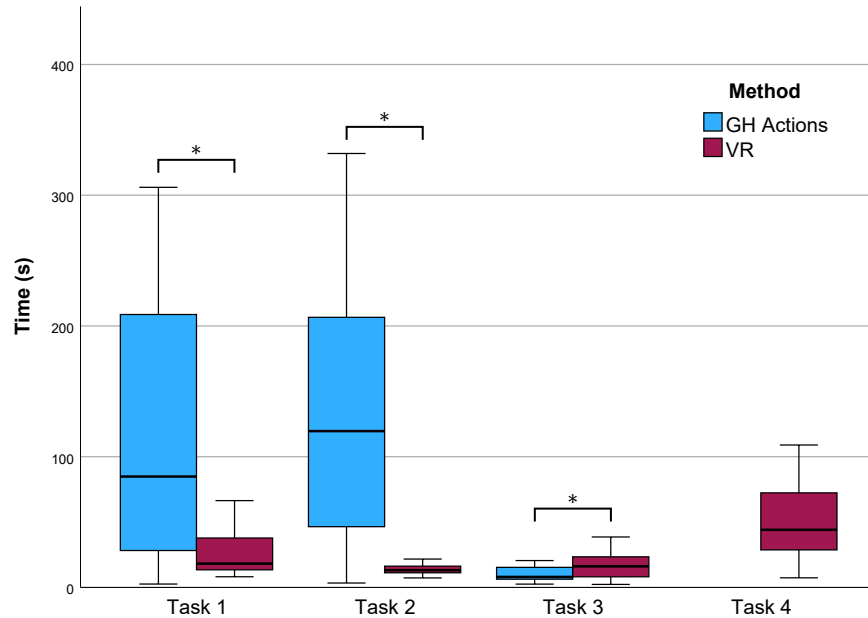


Figure 5.5: Task completion time distributions across all evaluation tasks. Box plots show the median, quartiles, and outliers for each environment. The asterisk (*) indicates statistical significance.

Task 1: The VR interface demonstrated significantly superior performance for finding workflow run times. Participants located the time of the most recent "Update Cargo.lock" run substantially faster in VR (Median = 18.22s, IQR = 26.11s) compared to GitHub Actions (Median = 84.82s, IQR = 191.52s). The Wilcoxon signed-rank test confirmed this difference as statistically significant ($Z = -3.323$, $p < .001$), indicating that the spatial timeline representation in VR provides more efficient access to temporal information than GitHub Actions' chronological listing approach.

Task 2: VR showed significant advantages for locating specific failed runs and their IDs. Participants found the most recent failed "CI" run and identified its ID considerably faster using the VR interface (Median = 13.17s, IQR = 6.01s) versus GitHub Actions (Median = 119.58s, IQR = 171.12s). The paired t-test analysis revealed a statistically significant difference ($t(18) = 5.179$, $p < .001$), suggesting that the immersive color-coding and spatial organization in VR facilitates rapid identification of failed runs and access to their information compared to traditional web-based interfaces.

Task 3: Contrary to Tasks 1 and 2, GitHub Actions demonstrated superior performance for identifying specific failed jobs. Participants found the jobs that caused the Task 2 run failure and identified their names faster using GitHub Actions (Median = 8.04s, IQR = 11.12s) compared to

VR (Median = 16.16s, IQR = 16.17s). The paired t-test confirmed statistical significance ($t(17) = -2.740$, $p = .015$), indicating that traditional interfaces may provide advantages for drilling down into specific job details within failed runs.

Task 4: As a VR-exclusive task testing spatial audio effectiveness, participants used only audio cues to locate and demonstrate a failing run. Task completion times (Median = 44.10s, IQR = 49.74s) provide performance metrics for audio-only failure detection in immersive environments.

Task Success Rates

Success rate analysis revealed advantages for the VR interface across most evaluation tasks. Figure 5.6 illustrates the success rate distributions across tasks and environments.

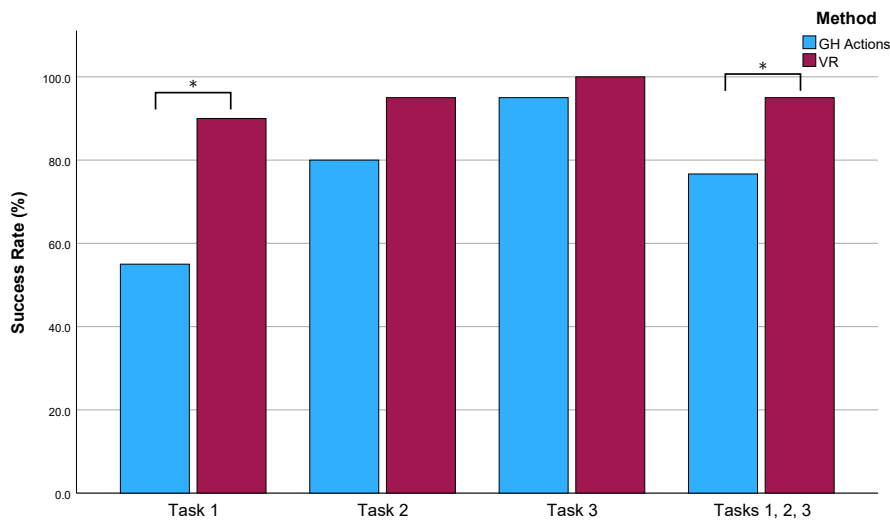


Figure 5.6: Task success rates by environment. The asterisk (*) indicates statistical significance.

Task 1: VR demonstrated significantly higher success rates (90.0%) compared to GitHub Actions (55.0%). McNemar's test confirmed this difference as statistically significant ($\chi^2 = 5.143$, $p = 0.016$), suggesting that the spatial timeline representation reduces navigation errors and improves task completion reliability.

Task 2: Both interfaces achieved high success rates, with VR slightly higher (95.0%) than GitHub Actions (80.0%). However, McNemar's test indicated no statistically significant difference ($\chi^2 = 0.800$, $p = 0.375$), suggesting comparable effectiveness for locating failed runs and identifying their IDs once participants navigate to the relevant workflow.

Task 3: Both environments achieved near-perfect success rates (GitHub Actions: 95.0%, VR: 100.0%) with no statistically significant difference ($\chi^2 = 0.000$, $p = 1.000$). This indicates that both interfaces adequately support identifying failed jobs once users navigate to the appropriate failed run.

Aggregated Analysis: Across all comparative tasks (Tasks 1-3), VR demonstrated significantly superior overall success rates (95.0%) compared to GitHub Actions (76.7%). The Wilcoxon

signed-rank test confirmed statistical significance ($Z = 51.000$, $p = 0.013$), indicating the systematic advantages of the immersive interface for CI/CD pipeline analysis tasks.

Task 4: The spatial audio-only navigation achieved a 65.0% success rate, providing effectiveness metrics for multimodal interface components.

Performance Summary

VR demonstrates clear superiority for overview tasks requiring temporal navigation and pattern recognition (Tasks 1 and 2), while GitHub Actions maintains advantages for detailed information retrieval (Task 3). The consistently higher success rates in VR suggest improved user confidence and reduced error rates, particularly for complex navigation scenarios.

5.5.2 System Usability Evaluation

System usability assessment revealed significant user preference for the VR interface over GitHub Actions. The System Usability Scale (SUS) scores were calculated using the standard approach: raw item scores were first converted to adjusted scores ranging from 0 (poorest rating) to 4 (best rating). For odd-numbered items (1, 3, 5, 7, 9), 1 is subtracted from the raw score; for even-numbered items (2, 4, 6, 8, 10), the raw score is subtracted from 5. The sum of adjusted scores is then multiplied by 2.5 to obtain the final SUS score ranging from 0 to 100 [27]. Figure 5.7 presents the SUS score distributions for both systems.

The VR interface achieved significantly higher SUS scores (Mean = 78.0, SD = 10.44, Median = 80.0, IQR = 12.50) compared to GitHub Actions (Mean = 56.38, SD = 19.29, Median = 58.75, IQR = 35.63). The paired t-test confirmed this difference as statistically significant ($t(19) = -4.951$, $p < .001$). The difference suggests that participants found the immersive visualization approach more intuitive and user-friendly despite being a novel interface paradigm.

Individual SUS Components

Analysis of individual SUS questionnaire items revealed specific usability advantages and challenges for each interface. Table 5.3 presents the detailed comparison across all SUS dimensions.

The VR system demonstrated significant advantages in several key usability dimensions. Participants found VR less complex (SUS 2) and easier to use (SUS 3), with an easier learning curve (SUS 7) and higher confidence levels (SUS 9). The VR interface was also perceived as more consistent (SUS 6), less cumbersome (SUS 8), and having better function integration (SUS 5) compared to GitHub Actions.

No significant differences were found for frequency of use intentions (SUS 1), technical support needs (SUS 4), and learning requirements (SUS 10).

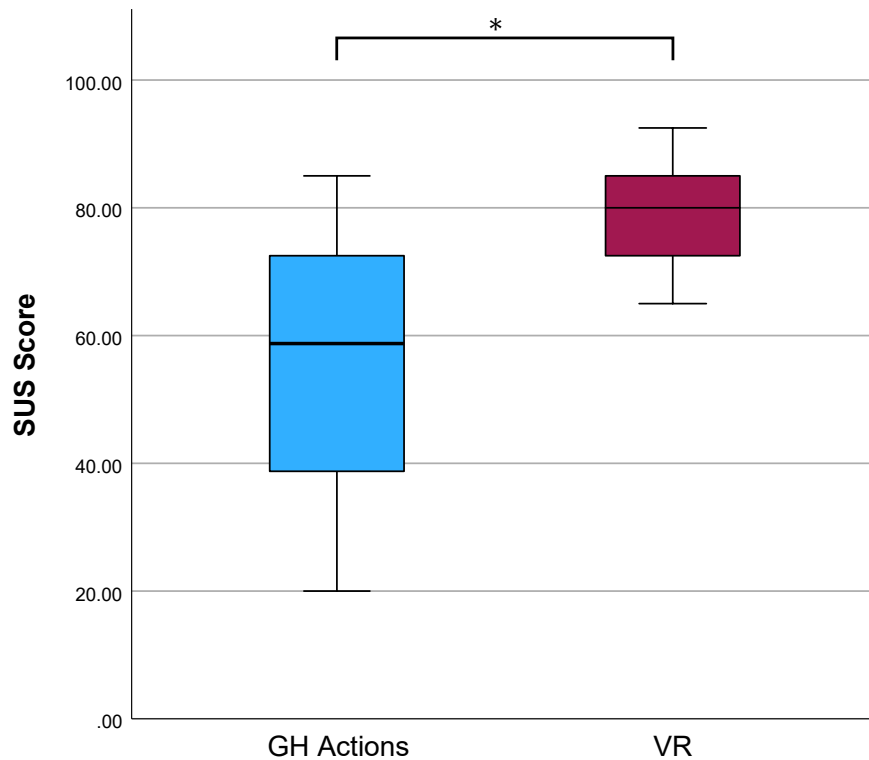


Figure 5.7: SUS score comparison between GitHub Actions and VR interfaces. The asterisk (*) indicates statistical significance.

5.5.3 Spatial Audio Assessment (Task 4)

Task 4 exclusively evaluated the effectiveness of spatial audio feedback for failure detection in the VR environment using audio-only navigation with all visual status indicators disabled.

User Confidence and Perceptions

A evaluation of the effectiveness of spatial audio showed that users had moderate confidence in its performance and found it to be somewhat useful. Table 5.4 presents participant responses to spatial audio evaluation questions.

These results indicate that while spatial audio cues were detectable, they required concentrated attention for effective interpretation. Participants perceived audio-only navigation as moderately mentally demanding and expressed uncertainty about the reliability of audio-only failure detection. When compared to visual indicators, spatial audio was rated as less effective, suggesting that audio serves better as a supplementary rather than a primary feedback channel.

Performance Metrics

Figure 5.8 presents the relationship between user confidence levels and actual task success rates.

Table 5.3: Median (IQR) and Wilcoxon Signed Ranks test results for individual SUS questionnaire items. The asterisk (*) indicates a significant difference between GitHub Actions and VR interfaces.

SUS Item	Median (IQR)		Wilcoxon Signed Ranks
	GitHub Actions	VR	
SUS 1: I think that I would like to use this system frequently.	4 (2)	4 (2)	$Z = -0.321, p = 0.748$
SUS 2: I found the system unnecessarily complex. *	3 (2)	2 (1)	$Z = -2.950, p = 0.003$
SUS 3: I thought the system was easy to use. *	3 (2)	4.5 (1)	$Z = -3.719, p < 0.001$
SUS 4: I think that I would need the support of a technical person to be able to use this system.	2 (2)	2 (2)	$Z = -1.102, p = 0.271$
SUS 5: I found the various functions in this system were well integrated. *	3.5 (3)	4 (1)	$Z = -2.176, p = 0.030$
SUS 6: I thought there was too much inconsistency in this system. *	2 (2)	1 (1)	$Z = -2.500, p = 0.012$
SUS 7: I would imagine that most people would learn to use this system very quickly. *	3 (2)	4 (2)	$Z = -2.678, p = 0.007$
SUS 8: I found the system very cumbersome to use. *	3 (2)	1.5 (1)	$Z = -3.461, p < 0.001$
SUS 9: I felt very confident using the system. *	2.5 (2)	4 (1)	$Z = -3.039, p = 0.002$
SUS 10: I needed to learn a lot of things before I could get going with this system.	2.5 (3)	2 (2)	$Z = -1.826, p = 0.068$

Table 5.4: Median (IQR) responses to spatial audio evaluation questions on 5-point Likert scales.

Question	Median (IQR)
How easy was it for you to figure out which direction the sound of the fault was coming from?	3 (2)
How mentally demanding was it to locate the fault based on the sound alone?	3 (2)
How confident were you that you had found the faulty run correctly?	3 (3)
Compared to using visual clues (colors on the runs and jobs), how do you rate the effectiveness of sound?	2 (2)

The spatial audio system achieved an overall success rate of 65% (13 successful completions out of 20 attempts). Performance varied considerably across confidence levels: participants expressing confidence level 1 achieved 100% success (2/2), confidence level 2 showed 16.7% success (1/6), confidence level 3 achieved 100% success (3/3), confidence level 4 maintained 100% success (4/4), and confidence level 5 reached 60% success (3/5).

Task completion times for successful audio navigation ranged from 7.37s to 106.34s (Median = 44.10s, IQR = 49.74s), indicating substantial variability in spatial audio interpretation efficiency among participants.

No significant correlation was found between user confidence levels and actual task success (Pearson $r = 0.184, p = 0.436$; Spearman $\rho = 0.187, p = 0.430$), indicating that subjective confidence did not reliably predict spatial audio navigation performance.

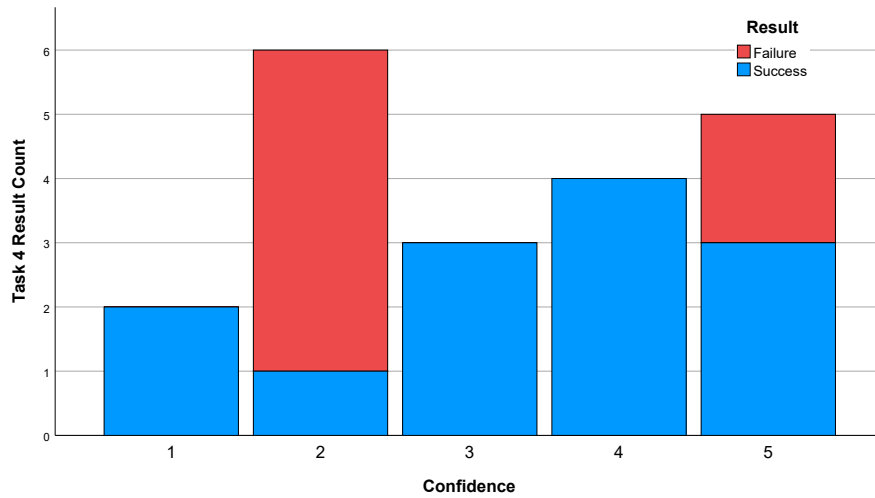


Figure 5.8: Task 4 success and failure distribution by user confidence levels. Results show the relationship between participant confidence in spatial audio navigation and actual task performance.

5.5.4 Experience Correlation Analysis

Analysis of participant experience levels revealed minimal correlation between prior technical experience and both system usability ratings and task completion times. For GitHub Actions, neither GitHub Actions experience ($r = 0.319$, $p = 0.171$) nor CI/CD pipeline experience ($r = 0.322$, $p = 0.166$) showed significant correlation with SUS scores. Similarly, VR system usability was not significantly correlated with VR experience ($\rho = 0.055$, $p = 0.818$) or CI/CD experience ($\rho = 0.224$, $p = 0.341$).

Task completion time analysis showed similar patterns. For GitHub Actions, no significant correlations were found between experience levels and total task completion times (GitHub Actions experience: $r = -0.091$, $p = 0.728$; CI/CD experience: $r = -0.048$, $p = 0.856$). VR performance also showed no significant correlation with experience levels (VR experience: $r = -0.263$, $p = 0.291$; CI/CD experience: $r = 0.035$, $p = 0.891$). Figures 5.9 and 5.10 present heatmap visualizations examining these relationships.

The heatmap analyses consistently show that most participants had limited VR experience, while other experience levels were more distributed. Although the visualizations suggest potential trends, the small sample size within each experience combination prevents drawing reliable conclusions. The limited sample distribution reinforces the statistical findings, indicating that experience levels did not significantly influence either usability perceptions or task performance for either interface.

5.5.5 Qualitative Observations

Throughout the evaluation sessions, participant feedback revealed specific usability insights and interface-specific challenges that provide context for the quantitative findings.

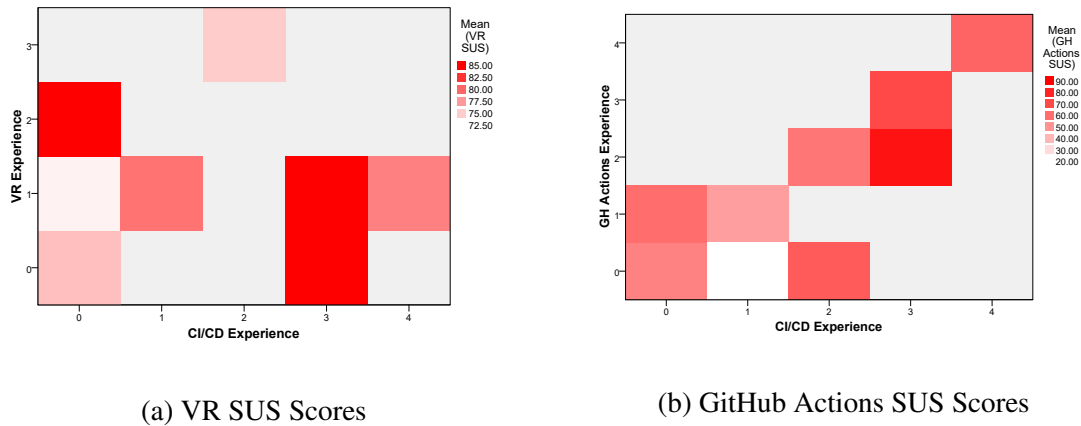


Figure 5.9: Heatmaps showing mean SUS scores across different experience level combinations. (a) VR experience vs CI/CD experience for VR interface ratings. (b) GitHub Actions experience vs CI/CD experience for GitHub Actions ratings.

GitHub Actions Interface Feedback

Participant feedback highlighted several usability challenges with GitHub Actions' interface design. Users consistently noted that "finding hours in GitHub is not very intuitive" and that "there are things that are not intuitive to access and could be more visible like the time when the run failed." Several participants emphasized the learning curve required, with one noting: "I felt it was a situation that required some learning about where things are before using, since it's not intuitive."

Even experienced users found specific tasks challenging: "Despite using GitHub Actions daily and thinking it's very useful and easy to use, I didn't find the way to discover the run start time and its ID intuitive." Participants noted visual confusion: "Visually, GitHub Actions are very confusing compared to the product that shows the latest runs in a more intuitive way."

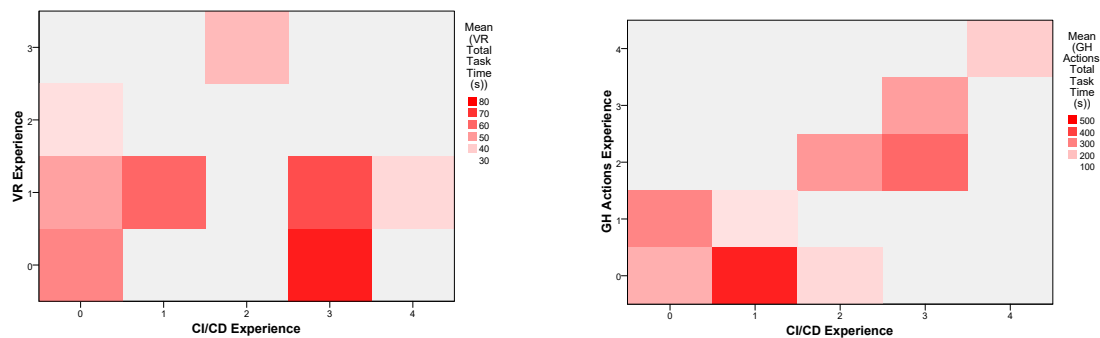
However, participants acknowledged the tool's value: "Having some experience with Git Actions, I think it's an ideal tool for everyone who wants to work on GitHub to ensure that what is done runs well and passes all tests without having to manually run the tests."

These feedback patterns aligned with observed technical challenges: timestamps were only visible on hover without visual indication, and workflow run IDs required indirect navigation to access.

VR Interface Feedback

Participants expressed strong appreciation for the spatial organization and visual design. One participant noted: "I really liked visualizing the workflows, could have a feature to show even more details within a workflow, like jobs and their states." The color-coding system received particularly positive feedback: "Colors help immensely to identify the status of runs and jobs because even being farther away you can identify them."

Users found the interface intuitive once familiar with controls: "After understanding how the commands worked and what they did, it became easier to interact with the environment and



(a) VR total task completion times

(b) GitHub Actions total task completion times

Figure 5.10: Heatmaps showing mean total task completion times across different experience level combinations. (a) VR experience vs CI/CD experience for VR interface task times. (b) GitHub Actions experience vs CI/CD experience for GitHub Actions task times.

understand how to achieve the task objectives." The overall design was well-received: "I found the product quite easy to use and intuitive. It gives a good temporal perspective of events and also of runs with failures."

However, participants suggested improvements for detailed information access: "I think it would make sense to have a dropdown appear when pointing to a particular job, not just for the entire run, because otherwise it's not easy to know the name of the failed job, why it failed, etc."

Spatial Audio System Feedback

Participants provided mixed feedback about the spatial audio implementation. The most common challenge was vertical localization: "The vertical difference hindered the identification of the sound direction" and "I didn't perceive where the sound came from." Several noted: "The sound seemed very similar on floors 0 and 1, but seemed a little louder on 1."

Despite challenges, participants saw potential: "It would have been easier if I knew beforehand how something with a fault sounds" and "Although I didn't get it right, the sound is much more intuitive." Users suggested multimodal enhancements: "The combination of visual feedback (e.g., intermittent light) with sound can increase the effectiveness of identifying problems associated with runs."

These qualitative insights support the quantitative findings, suggesting that VR excels at overview and pattern recognition tasks, while traditional interfaces maintain advantages for detailed information consumption. The feedback highlights specific areas for future development, particularly in spatial audio design and detailed information presentation.

Limitations

Several limitations should be considered when interpreting these results:

- **Sample Size:** With 20 participants, the study provides adequate power for detecting large effects but may miss smaller differences between interfaces.
- **Participant Demographics:** The predominantly young, academic sample may not represent all developer populations, particularly experienced industry professionals.
- **Task Scope:** The evaluation focused on specific CI/CD analysis tasks and may not generalize to all workflow management activities.
- **Temporal Constraints:** The fixed dataset approach, while ensuring consistency, may not capture the dynamic nature of real-world CI/CD environments.
- **Learning Effects:** Despite counterbalancing, some residual learning effects between environments may influence results.

5.6 Discussion

The evaluation results demonstrate clear performance and usability advantages for the VR-based CI/CD visualization system across most evaluation dimensions. The immersive 3D interface showed superior performance for temporal navigation and failure detection tasks while achieving significantly higher user satisfaction scores. However, traditional interfaces maintained advantages for detailed information retrieval tasks, revealing complementary strengths between visualization approaches.

Performance analysis revealed distinct patterns across task categories. The VR system excelled at overview tasks requiring spatial and temporal pattern recognition, with participants completing temporal navigation (Task 1) and failure detection (Task 2) significantly faster than with GitHub Actions. Conversely, GitHub Actions demonstrated superior performance for job-level debugging (Task 3), suggesting that traditional interfaces remain effective for detailed information consumption. The spatial audio system achieved moderate success rates (65%) for audio-only failure detection, indicating potential for multimodal feedback while highlighting implementation challenges.

Usability assessment showed strong user preference for the VR interface, with SUS scores averaging 21.6 points higher than GitHub Actions. Individual SUS components revealed that participants found the VR system less complex, easier to use, and more confidence-inspiring despite being a novel interface paradigm. Qualitative feedback supported these findings, with users appreciating the spatial organization and color-coding while suggesting improvements for detailed information access.

Experience correlation analysis revealed that prior technical background did not significantly influence performance or usability perceptions for either interface. This finding suggests that the VR system's advantages are not dependent on specialized knowledge, supporting its potential accessibility to users with varying experience levels.

Taking everything into account, the research questions can be addressed as follows:

RQ1 Does a VR-based visualization system help users more easily identify and interpret CI/CD workflow structure and failures compared to a traditional 2D interface? The results strongly support the effectiveness of VR for workflow structure understanding and failure identification. Participants achieved significantly higher success rates in VR across aggregated tasks (95.0% vs 76.7%, $p = 0.013$), with particularly pronounced advantages for temporal navigation (90.0% vs 55.0%, $p = 0.016$) and failure detection tasks.

The spatial timeline representation in VR enabled participants to locate workflow run times substantially faster (Median = 18.22s vs 84.82s, $p < .001$) and identify failed runs more efficiently (Median = 13.17s vs 119.58s, $p < .001$). Qualitative feedback reinforced these findings, with participants noting that "colors help immensely to identify the status of runs and jobs because even being farther away you can identify them" and appreciating the "good temporal perspective of events." These results indicate that the 3D spatial layout and immersive visualization significantly improve users' ability to understand workflow structures and rapidly identify failures compared to traditional 2D interfaces.

RQ2 Does using spatial layout in a 3D environment contribute to reduced task time and fewer errors when exploring workflow execution data? The evaluation provides mixed but generally supportive evidence for this question. For overview and navigation tasks, the spatial 3D layout demonstrated clear advantages in both completion times and error reduction. Tasks 1 and 2 showed dramatically reduced completion times in VR, with the spatial timeline enabling participants to navigate temporal information more efficiently than GitHub Actions' chronological listing approach.

However, Task 3 revealed limitations of the spatial approach for detailed information retrieval, where GitHub Actions outperformed VR (Median = 8.04s vs 16.16s, $p = .015$). This suggests that while spatial layout excels for overview tasks and pattern recognition, traditional interfaces may be more efficient for drilling down into specific textual details. The consistently higher success rates across aggregated tasks (95.0% vs 76.7%) demonstrate that spatial layout does contribute to error reduction overall, particularly for complex navigation scenarios that require understanding relationships between workflow components.

RQ3 How do users perceive the ease of use, learnability, and overall satisfaction with the VR-based visualization system compared to traditional tools? User perceptions strongly favored the VR system across multiple usability dimensions. SUS scores were significantly higher for VR (Mean = 78.0) compared to GitHub Actions (Mean = 56.38, $p < .001$), indicating substantially better-perceived usability despite VR being a novel interface paradigm. Individual SUS analysis revealed that participants found VR significantly easier to use ($p < .001$), less complex ($p = 0.003$), and more confidence-inspiring ($p = 0.002$). The learning curve was perceived as easier ($p = 0.007$), with better function integration ($p = 0.030$) and greater consistency ($p = 0.012$). Participants felt the VR system was less cumbersome to use

($p < .001$) compared to GitHub Actions. No significant differences were found for technical support needs or learning requirements, suggesting comparable initial learning barriers.

Qualitative feedback supported these quantitative findings, with participants describing the VR system as "easy to use and intuitive" and noting that "after understanding how the commands worked and what they did, it became easier to interact with the environment." However, participants also identified areas for improvement, particularly requesting enhanced, detailed information access: "It would make sense to have a dropdown appear when pointing to a particular job, not just for the entire run."

The spatial audio system received moderate acceptance, with participants rating it as less effective than visual cues but recognizing its potential as supplementary feedback. Users suggested multimodal enhancements combining visual and audio feedback to improve effectiveness.

These findings demonstrate that VR-based visualization can significantly improve CI/CD workflow analysis for overview and pattern recognition tasks while maintaining comparable learning requirements to traditional interfaces.

Compared to existing work like VR-DevOps, this research offers advantages: a more intuitive factory metaphor versus abstract spatial layouts, temporal compression algorithms for handling real-world data scale, multimodal feedback through spatial audio, and empirical validation through controlled user studies. While VR-DevOps demonstrated the feasibility of immersive pipeline visualization, it lacked scalability solutions and user-centered validation that this work can provide. Additionally, the VR-DevOps implementation is not publicly available, limiting the possibility of direct comparison. These differences made direct comparison impractical, as the systems address different aspects of the immersive CI/CD visualization challenge.

The results suggest that immersive visualization represents a promising direction for developer tools, particularly for complex data exploration scenarios where spatial relationships and temporal patterns are important. However, successful implementation requires careful consideration of when to use spatial versus traditional presentation modes, with hybrid approaches potentially offering optimal user experiences.

Chapter 6

Conclusions

This dissertation investigated how immersive Virtual Reality (VR) environments can enhance the visualization and understanding of CI/CD pipelines, addressing the limitations of traditional 2D interfaces for exploring complex pipeline structures, temporal patterns, and failure analysis. While this study used GitHub Actions as the representative traditional interface, the findings can be generalized to other 2D CI/CD platforms.

This research contributes to the intersection of immersive analytics and software engineering visualization by introducing a framework for translating CI/CD pipeline semantics into immersive 3D environments. The factory metaphor effectively represents abstract workflow concepts: workflows as floors, workflow runs as blocks, jobs as sub-blocks, and time as spatial depth. A fully functional VR prototype was developed using Unity and the XR Interaction Toolkit¹, featuring temporal-to-spatial mapping algorithms, lane assignments for simultaneous executions, time compression portals for managing inactive periods, and spatial audio feedback for failure detection.

Through a user study with 20 participants using standardized tasks and validated measurement instruments, the research establishes evidence-based guidelines for immersive CI/CD visualization, demonstrating that spatial 3D layouts enhance overview tasks, pattern recognition, and temporal navigation. At the same time, traditional interfaces remain superior for detailed information retrieval and textual analysis.

6.1 Key Findings

The evaluation highlighted significant advantages of VR-based visualization for specific tasks while emphasizing important contexts where traditional interfaces are more effective. The findings address the research questions outlined at the beginning of this study.

Research Question 1 asked whether VR-based visualization helps users more easily identify and interpret CI/CD workflow structure and failures compared to traditional 2D interfaces. The results support this hypothesis, with participants achieving significantly higher success rates in

¹Unity XR Interaction Toolkit Documentation: <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@3.0/manual/index.html>

VR and faster completion times for temporal navigation and failure detection tasks. The spatial timeline representation and immersive color coding proved effective for rapid pattern recognition and failure identification.

Research Question 2 examined whether spatial layout in 3D environments contributes to reduced task time and fewer errors. The evidence provides mixed but generally supportive findings. While VR demonstrated clear advantages for overview and navigation tasks, traditional interfaces maintained superiority for detailed information retrieval tasks. This suggests that spatial layout excels for high-level reasoning but may introduce overhead for fine-grained analysis.

Research Question 3 investigated user perceptions of ease of use, learnability, and satisfaction. Despite being a novel interface paradigm, the VR system achieved significantly higher usability scores, with participants finding it less complex, easier to use, and more confidence-inspiring. Importantly, no significant differences were found in learning requirements, suggesting that the VR system's advantages are accessible to users with varying experience levels.

Additional Insights: The spatial audio system achieved moderate effectiveness for failure detection, indicating potential for multimodal interfaces while highlighting implementation challenges, particularly in vertical sound localization.

6.2 Future Work

Several promising directions emerge from this research for future investigation and development:

Enhanced Information Access and Interaction

Users expressed interest in accessing more detailed information about individual jobs and pipeline components. Future implementations could provide hierarchical information displays where pointing at runs shows basic information, but drilling into specific jobs reveals detailed logs, execution parameters, and performance metrics. This could be achieved through contextual menus, gesture-based interactions, or voice commands that maintain the immersive experience while providing the granular access users expect from traditional CI/CD interfaces.

Improved Spatial Audio and Multimodal Feedback

The moderate effectiveness of spatial audio suggests significant room for improvement. Future work could explore enhanced 3D audio algorithms, personalized audio calibration, and multimodal feedback combining visual and auditory cues. Additionally, addressing the identified challenges with vertical sound localization could involve implementing head-related transfer functions (HRTFs) [3] or alternative audio positioning techniques that effectively utilize the spatial capabilities of virtual reality.

Visual and Aesthetic Improvements

While the block-based visualization effectively communicates pipeline structure, future work could develop more visually engaging factory representations that better capture the metaphor's potential. This could include animated conveyor belts that move runs and jobs through the pipeline during execution, factory machinery that responds to different job types, and environmental elements like steam, lighting effects, or mechanical sounds that enhance the factory atmosphere.

Temporal Navigation and Filtering

Current spatial navigation requires users to physically walk through time, which can be cumbersome for exploring distant time periods. Future systems could implement temporal teleportation features, allowing users to jump directly to specific dates, times, or pipeline events. Advanced filtering capabilities enable users to show only failed runs, filter by specific time ranges, or focus on particular workflow types. These features could be controlled through voice commands ("show me all failures from last week") or gesture-based timeline scrubbing, making temporal exploration more efficient while maintaining spatial context.

Advanced Analytics and Pattern Recognition

Building on user feedback about understanding performance trends, future work could incorporate features for aggregating and analyzing multiple runs from the same workflow. This might include visual indicators for performance degradation over time, automatic clustering of similar failure patterns, or predictive highlighting of runs likely to fail based on historical data. Machine learning integration could automatically surface anomalies, identify recurring issues, and suggest optimization opportunities, transforming the visualization from a passive display into an active analytical tool.

Real-Time Data Integration and Dynamic Visualization

Extending the system to handle live CI/CD data streams presents technical and design challenges. Real-time integration would require performance optimization, incremental visualization updates, and robust error handling to maintain smooth VR experiences. The visual representation could become more engaging with runs appearing and moving through the factory as they execute, with different speeds and visual effects based on job types and execution status. Implementing efficient data streaming, level-of-detail rendering, and predictive loading could enable visualization of larger datasets while supporting lower-end VR hardware.

Collaborative Features and Multi-User Support

Multi-user support would enable collaborative debugging sessions and team-based CI/CD analysis. This could include shared virtual spaces where team members can point out issues, leave virtual annotations, and collaboratively explore pipeline problems. Features like shared bookmarks,

collaborative filtering, and real-time voice communication could make the VR environment viable for distributed teams to analyze and troubleshoot CI/CD issues together.

Cross-Platform and Accessibility Enhancements

Extending the approach to other CI/CD platforms (Jenkins, GitLab CI, Azure DevOps) would validate the generalizability of the factory metaphor and enable broader adoption. Additionally, accessibility improvements could include adaptations for users with visual, auditory, or motor impairments, alternative interaction methods for users uncomfortable with VR, and seamless integration with existing development workflows and tools.

6.3 Concluding Remarks

This research demonstrates that immersive Virtual Reality significantly benefits CI/CD workflow analysis, especially in spatial-temporal reasoning and pattern recognition, while it acknowledges that traditional interfaces are still better for specific detailed tasks. The factory metaphor and empirical validation provide a foundation for hybrid approaches that thoughtfully integrate immersive and traditional visualization techniques. As VR technology becomes more accessible, the principles demonstrated here extend beyond CI/CD systems to broader questions about how immersive technologies can enhance human understanding of complex technical systems, making this work relevant across multiple domains within computer science and software engineering.

References

- [1] Azeem Ahmad, Ola Leifler, and Kristian Sandahl. Data visualisation in continuous integration and delivery: Information needs, challenges, and recommendations. *IET Software*, 16(3):331–349, 2022.
- [2] Ahmed Bayyari and M. Eduard Tudoreanu. The impact of immersive virtual reality displays on the understanding of data visualization. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '06, page 368–371, New York, NY, USA, 2006. Association for Computing Machinery.
- [3] begault durand r., wenzel elizabeth m., and anderson mark r. direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source. *journal of the audio engineering society*, 49:904–916, october 2001.
- [4] Markus Berger and Ralf Bill. Combining VR Visualization and Sonification for Immersive Exploration of Urban Noise Standards. *Multimodal Technologies and Interaction*, 3(2):34, June 2019.
- [5] Lutz Bornmann. The sound of science. *EMBO reports*, 25(9):3743–3747, September 2024.
- [6] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D³ Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, December 2011.
- [7] D.A. Bowman, D. Koller, and L.F. Hodges. Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques. In *Proceedings of IEEE 1997 Annual International Symposium on Virtual Reality*, pages 45–52, March 1997.
- [8] Paul Bremner, Thomas J. Mitchell, and Verity McIntosh. The impact of data sonification in virtual reality robot teleoperation. *Frontiers in Virtual Reality*, 3, August 2022.
- [9] Steve Bryson. Virtual reality in scientific visualization. *Commun. ACM*, 39(5):62–71, May 1996.
- [10] Stuart K. Card, Jock Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, January 1999.
- [11] Maxime Cordeil, Andrew Cunningham, Benjamin Bach, Christophe Hurter, Bruce H. Thomas, Kim Marriott, and Tim Dwyer. IATK: An Immersive Analytics Toolkit. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 200–209, March 2019.

- [12] Carolina Cruz-Neira, Daniel J. Sandin, Thomas A. DeFanti, Robert V. Kenyon, and John C. Hart. The CAVE: Audio visual experience automatic virtual environment. *Commun. ACM*, 35(6):64–72, June 1992.
- [13] Hugo da Gíão, André Flores, Rui Pereira, and Jácome Cunha. Chronicles of CI/CD: A Deep Dive into its Usage Over Time, February 2024.
- [14] Steve Dübel, Martin Röhlig, Heidrun Schumann, and Matthias Trapp. 2D and 3D presentation of spatial data: A systematic review. In *2014 IEEE VIS International Workshop on 3DVis (3DVis)*, pages 11–18, November 2014.
- [15] Barrett Ens, Benjamin Bach, Maxime Cordeil, Ulrich Engelke, Marcos Serrano, Wesley Willett, Arnaud Prouzeau, Christoph Anthes, Wolfgang Büschel, Cody Dunne, Tim Dwyer, Jens Grubert, Jason H. Haga, Nurit Kirshenbaum, Dylan Kobayashi, Tica Lin, Monsurat Olaosebikan, Fabian Pointecker, David Saffo, Nazmus Saquib, Dieter Schmalstieg, Danielle Albers Szafir, Matt Whitlock, and Yalong Yang. Grand Challenges in Immersive Analytics. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, pages 1–17, New York, NY, USA, May 2021. Association for Computing Machinery.
- [16] Adrien Fonnet and Yannick Prié. Survey of Immersive Analytics. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2101–2122, March 2021.
- [17] Lucrezia Guiotto Nai Fovino, Anita Zanella, and Massimo Grassi. Evaluation of the effectiveness of sonification for time series data exploration, February 2024.
- [18] J. Friedl-Knirsch, F. Pointecker, S. Pfistermüller, C. Stach, C. Anthes, and D. Roth. A Systematic Literature Review of User Evaluation in Immersive Analytics. *Computer Graphics Forum*, 43(3):e15111, 2024.
- [19] Jiayi Hong, Rostyslav Hnatyshyn, Ebrar A. D. Santos, Ross Maciejewski, and Tobias Isenberg. A Survey of Designs for Combined 2D+3D Visual Representations. *IEEE Transactions on Visualization and Computer Graphics*, 30(6):2888–2902, June 2024.
- [20] Helen H Huang, Hanspeter Pfister, and Yalong Yang. Is embodied interaction beneficial? A study on navigating network visualizations. *Information Visualization*, 22(3):169–185, July 2023.
- [21] Lucas Joos, Maximilian T. Fischer, Julius Rauscher, Daniel A. Keim, Tim Dwyer, Falk Schreiber, and Karsten Klein. Visual Network Analysis in Immersive Environments: A Survey, January 2025.
- [22] Timothy Kinsman, Mairieli Wessel, Marco A. Gerosa, and Christoph Treude. How Do Software Developers Use GitHub Actions to Automate Their Workflows? In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 420–431, May 2021.
- [23] Joseph Kotlarek, Oh-Hyun Kwon, Kwan-Liu Ma, Peter Eades, Andreas Kerren, Karsten Klein, and Falk Schreiber. A Study of Mental Maps in Immersive Network Visualization. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 1–10, June 2020.
- [24] Matthias Kraus, Johannes Fuchs, Björn Sommer, Karsten Klein, Ulrich Engelke, Daniel Keim, and Falk Schreiber. Immersive Analytics with Abstract 3D Visualizations: A Survey. *Computer Graphics Forum*, 41(1):201–229, 2022.

- [25] Benjamin Lee, Dave Brown, Bongshin Lee, Christophe Hurter, Steven Drucker, and Tim Dwyer. Data visceralization: Enabling deeper understanding of data using virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1095–1105, 2021.
- [26] Benjamin Lee, Xiaoyun Hu, Maxime Cordeil, Arnaud Prouzeau, Bernhard Jenny, and Tim Dwyer. Shared Surfaces and Spaces: Collaborative Data Visualisation in a Co-located Immersive Environment. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1171–1181, February 2021.
- [27] James R. Lewis. The System Usability Scale: Past, Present, and Future. *International Journal of Human–Computer Interaction*, 34(7):577–590, July 2018.
- [28] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22 140:55–55, 1932.
- [29] Shixia Liu, Weiwei Cui, Yingcai Wu, and Mengchen Liu. A survey on information visualization: Recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, December 2014.
- [30] Ana Isabel Martins, Ana Filipa Rosa, Alexandra Queirós, Anabela Silva, and Nelson Pacheco Rocha. European Portuguese Validation of the System Usability Scale (SUS). *Procedia Computer Science*, 67:293–300, January 2015.
- [31] Quinn McNemar. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157, June 1947.
- [32] Jules Moloney, Branka Spehar, Anastasia Globa, and Rui Wang. The affordance of virtual reality to enable the sensory representation of multi-dimensional data for immersive analytics: from experience to insight. *Journal of Big Data*, 5(1):53, 2018.
- [33] Roy Oberhauser. VR-DevOps: Visualizing and Interacting with DevOps Pipelines in Virtual Reality.
- [34] Payod Panda, Byungsoo Kim, Sana Behnam-Asl, and Elif Sener. Immersive Analytics: A User-Centered Perspective. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, VRST ’21, pages 1–3, New York, NY, USA, December 2021. Association for Computing Machinery.
- [35] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone Trees: Animated 3D visualizations of hierarchical information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’91, pages 189–194, New York, NY, USA, March 1991. Association for Computing Machinery.
- [36] Sk Golam Saroar and Maleknaz Nayebi. Developers’ Perception of GitHub Actions: A Survey Analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, EASE ’23, pages 121–130, New York, NY, USA, June 2023. Association for Computing Machinery.
- [37] Sk Golam Saroar and Maleknaz Nayebi. Developers’ perception of github actions: A survey analysis. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 121–130, 2023.

- [38] Arvind Satyanarayan, Dominik Moritz, Kanit Wongsuphasawat, and Jeffrey Heer. Vega-Lite: A Grammar of Interactive Graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):341–350, January 2017.
- [39] Mojtaba Shahin, Muhammad Ali Babar, and Liming Zhu. Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, 5:3909–3943, 2017.
- [40] S. S. SHAPIRO and M. B. WILK. An analysis of variance test for normality (complete samples)†. *Biometrika*, 52(3-4):591–611, December 1965.
- [41] Ronell Sicat, Jiabao Li, Junyoung Choi, Maxime Cordeil, Won-Ki Jeong, Benjamin Bach, and Hanspeter Pfister. DXR: A Toolkit for Building Immersive Data Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):715–725, January 2019.
- [42] Carson Sievert. *Interactive Web-Based Data Visualization with R, Plotly, and Shiny*. Chapman and Hall/CRC, New York, January 2020.
- [43] Zonglin Tian, Xiaorui Zhai, Gijs van Steenpaal, Lingyun Yu, Evanthia Dimara, Mateus Espadoto, and Alexandru Telea. Quantitative and Qualitative Comparison of 2D and 3D Projection Techniques for High-Dimensional Data. *Information*, 12(6):239, June 2021.
- [44] Christian Tominski, Heidrun Schumann, Gennady Andrienko, and Natalia Andrienko. Stacking-Based Visualization of Trajectory Attribute Data. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2565–2574, December 2012.
- [45] Yunhai Wang, Fubo Han, Lifeng Zhu, Oliver Deussen, and Baoquan Chen. Line Graph or Scatter Plot? Automatic Selection of Methods for Visualizing Trends in Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 24(2):1141–1154, February 2018.
- [46] David Gray Widder, Michael Hilton, Christian Kästner, and Bogdan Vasilescu. A conceptual replication of continuous integration pain points in the context of Travis CI. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, pages 647–658, New York, NY, USA, August 2019. Association for Computing Machinery.
- [47] Frank Wilcoxon. Individual Comparisons by Ranking Methods. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, pages 196–202. Springer, New York, NY, 1992.