Immersive Software Archaeology: Collaborative Exploration and Note Taking in Virtual Reality

Adrian Hoff
adho@itu.dk
IT University of Copenhagen, Denmark

Mircea Lungu
mlun@itu.dk
IT University of Copenhagen, Denmark

Christoph Seidl
chse@itu.dk
IT University of Copenhagen, Denmark

Michele Lanza
michele.lanza@usi.ch
Software Institute @ USI Lugano, Switzerland

ABSTRACT

Understanding software systems is a vital task, often undertaken by teams of engineers, for the development and maintenance of systems. Collaborative software visualization tools are essential in this context, yet they are limited. Existing tools, particularly in virtual reality, allow exploration but lack the crucial feature of note-taking, which is a significant limitation.

We present Immersive Software Archaeology (ISA), a virtual reality tool that enables engineering teams to collaboratively explore and comprehend software systems. Unique to ISA, it facilitates note-taking during exploration with virtual multimedia whiteboards that support freehand diagramming, audio recordings, and VR screenshots. Notes taken on these whiteboards are synchronized with an Integrated Development Environment (IDE), providing easy access to the results of a VR exploration while performing changes to the system's source code.

Video Demonstration—https://youtu.be/32EIpf4V3b4

CCS CONCEPTS

• **Software and its engineering** → **Software maintenance tools**; **Maintaining software**; *Software reverse engineering*.

KEYWORDS

Software Visualization, Software Comprehension, Collaborative Software Engineering, Virtual Reality

ACM Reference Format:
Adrian Hoff, Mircea Lungu, Christoph Seidl, and Michele Lanza. 2024. Immersive Software Archaeology: Collaborative Exploration and Note Taking in Virtual Reality. In *32nd IEEE/ACM International Conference on Program Comprehension (ICPC '24), April 15–16, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3643916.3644438

ICPC '24, April 15–16, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0586-1/24/04
https://doi.org/10.1145/3643916.3644438

1 INTRODUCTION AND RELATED WORK

Exploring and comprehending an unfamiliar software system, e.g., for re-engineering a legacy system, is a complex yet crucial task typically performed by teams of engineers [19, 23]. When done by reading through source code, the process is hampered the size and complexity of a system. Software visualization can be beneficial in this scenario: by using visual metaphors to represent the structure, behavior, or evolution of a system, it provides software engineers with a comprehensive overview [2, 6].

Software visualizations vary in their metaphor (e.g., graphs [7, 14] or information cities [11, 16, 21]), dimensionality (2D [15, 17] or 3D), and – for 3D visualizations – display medium (standard screen [20, 22, 24] or virtual/augmented reality (VR/AR) [4, 5, 9, 18]).

Despite the availability of many software visualization tools, there is a scarcity of collaborative options, especially for remote settings. VR is a preferred medium for such settings [12, 13], but a major drawback of existing VR visualizations is their lack of support for note-taking during exploration, risking the loss of insights.

We introduce Immersive Software Archaeology (ISA), a collaborative VR software visualization tool. Based on automated system analysis, ISA allows software engineering teams to collaboratively explore a system's structure in immersive VR using an interactive visualization that is synchronized over the internet. Engineers can record their thoughts and insights on collaborative multimedia whiteboards during their VR exploration - which is not possible with existing VR tools. Post-exploration, these notes are accessible in the Integrated Development Environment (IDE) Eclipse, aiding in the implementation of changes to the system's source code.

2 COLLABORATIVE SOFTWARE EXPLORATION IN VR

We present the collaborative virtual reality software visualization tool Immersive Software Archaeology (ISA) in its current version 2.1[1]. ISA is comprised of two main components: a model server that integrates with the widely used open-source development environment Eclipse, and a VR visualization client. We discuss how users interact with both components, and follow with an overview of the relevant aspects of its architecture.

2.1 Usage from a User's Point of view

Figure 1 offers an overview of ISA, illustrated through screenshots captured from the perspective of a user. The process of using ISA

[1] https://gitlab.com/immersive-software-archaeology

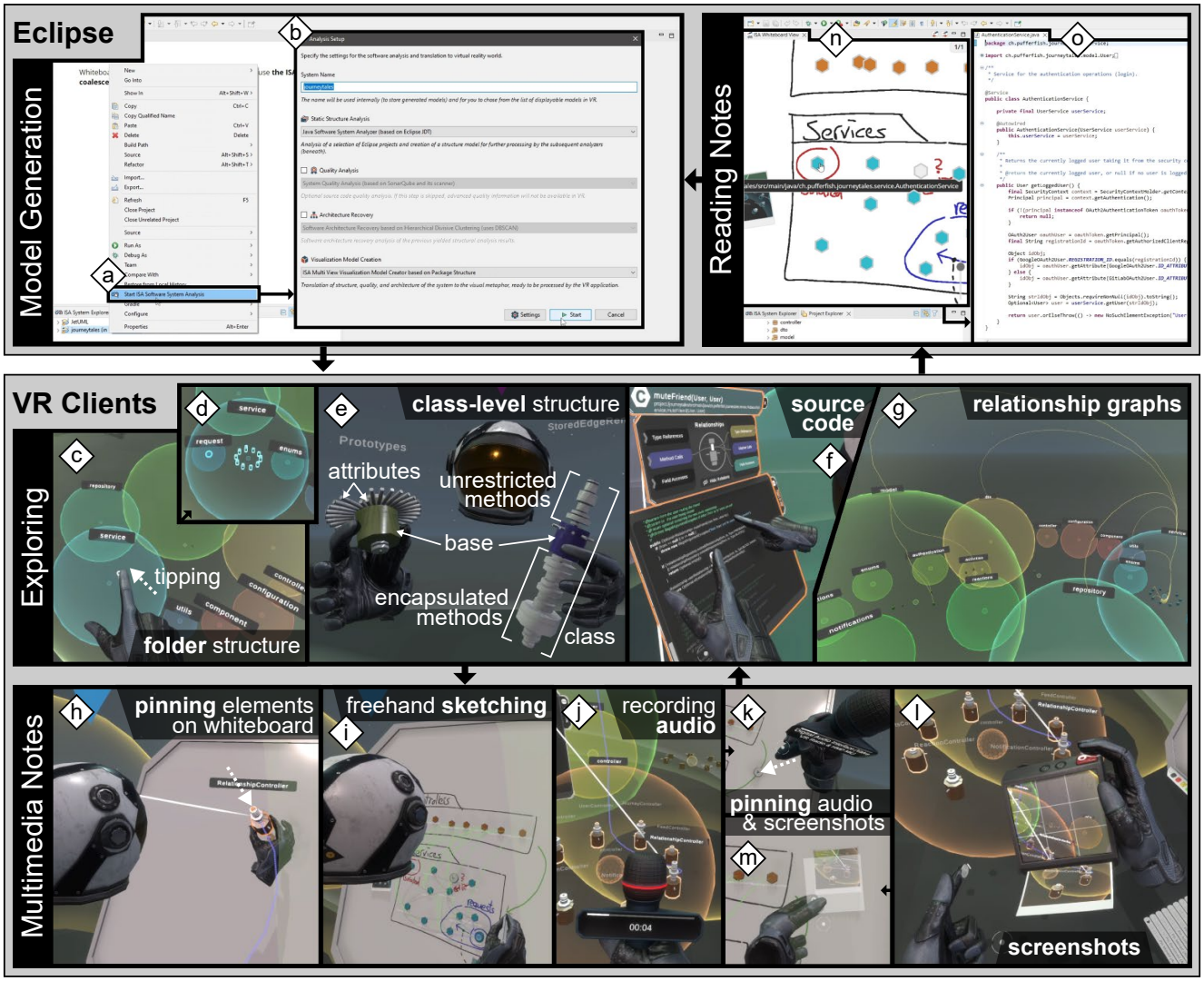

Figure 1: Overview of ISA's collaborative VR software exploration and note taking approach from a user's point of view.

begins within Eclipse, where users initiate an automated procedure to generate the information necessary to visualize a selected subject system (ⓐ and ⓑ). Once that is completed, multiple users are able to jointly explore the subject system in a collaborative VR visualization and record multimedia notes about their observations and insights (ⓒ to ⓙ). Finally, notes taken in VR are accessible in Eclipse, allowing users to review them while making modifications to the source code of the system under study, as indicated in ⓝ, ⓞ.

Automated Model Generation in Eclipse. Users start ISA's automated model generation process via an entry in the Eclipse project explorer's context menu ⓐ. This process involves multiple steps (see Section 2.2), for which users choose between alternative implementations depending on the ISA Eclipse plugins installed ⓑ.

Upon completion of the model generation process, the results are persisted and users make them available for VR visualization clients by launching ISA's model server, either through a confirmation dialog that pops up after the model generation process or by using a control panel view in the Eclipse UI.

Collaborative Exploration in Virtual Reality. Once the ISA Eclipse model server is reachable, multiple users can connect to it via a local network or the internet using the ISA VR client running on a head-mounted VR device. Once connected, they have the option to select and load a system from the range of those analyzed in the connected ISA Eclipse model server.

After the loading phase completes, users enter in a real-time, synchronized visualization environment of the selected system where they can view the virtual representations of fellow collaborators, observing each other's interactions with elements of the visualization as detailed in the subsequent sections.

Folder Spheres. ISA visualizes the folder-level structure of a subject system in form of nested semi-transparent folder spheres with different colors. These folder spheres are initially in a closed state,

as shown in ⓒ. Users can interact with them by tapping on them to open and reveal the contents within, compare ⓒ and ⓓ. This interaction promotes a top-down approach to exploring the system's structure [3]. Additionally, ISA enables users to utilize hand gestures to manipulate the scale of the visualization and to move it within the virtual space, providing an additional form of navigation besides the standard click-and-point VR teleportation mechanisms.

Class Cylinders. Positioned within folder spheres, ISA visualizes the class-level elements of a subject system as stacks of cylinders ⓔ consisting of four parts: (1) a base cylinder colored according to the containing folder sphere's color, (2) cylinders representing methods without access restrictions (e.g., public in Java), (3) cylinders representing encapsulated methods (e.g., private, protected, or package visible in Java), and (4) spikes originating from the base cylinder represent attributes.

The height of method cylinders is proportional to the number of expressions contained in the represented method while their radius is proportional to its cognitive complexity as measured by the metric proposed by Campbell ⓔ [1]. Attribute spikes vary in length depending on whether they are encapsulated (short spike) or accessible without restrictions (long spike).

With the above, ISA encodes a summary of structural class metrics into the shape of class cylinders and their constituents. Users can visually comprehend these already before reading code. Due to their cylindrical shape and symmetrical layout, class cylinders' outline is independent of the viewing angle, an aspect particularly relevant in a collaborative setting.

Relationship Edges. In ISA, tapping on the visual representations of folders, classes, methods, or attributes enables users to access a user interface that provides detailed information about the tapped element. This user interface is interactive and can be repositioned by the users as needed. All interactions with it are synchronized in real time among all users, enhancing the collaborative experience.

Figure 1 ⓕ depicts the user interface for a selected method. Its upper section features controls for a relationship visualization that represents references to or from a software element (e.g., a method) in the subject system's source code. It distinguishes between different types of references into type references, method calls, and field accesses. Users can dynamically display or hide these references for a selected element. For example, ⓖ shows type references originating from a selected class. The relationship visualization is available for all types of elements, including folders, where it aggregates references coming from or going to their constituent elements.

Source Code. When opening the user interface for an element containing direct source code (i.e., a class-level element, method, or attribute), it displays a scrollable view of the source code at its lower section, as shown in ⓕ. In combination, the relationship graph and code view provide a comprehensive and interactive means for users to explore and understand the code details and structural relationships between them in a subject system.

Collaborative Note Taking in Virtual Reality. Users can spawn virtual whiteboards to take notes during VR exploration sessions. These whiteboards allow for pinning elements from the visualization, creating freehand sketched diagrams, and attaching audio recordings as well as screenshots taken in VR.

Pinning Software Elements. Users can grab folder spheres and class cylinders from the visualization and pin them on a virtual whiteboard via a respective hand gesture (indicated in ⓗ). When an element is pinned to the whiteboard, a corresponding pin appears. Users can manipulate these pins by moving them around or accessing an interface with additional information, such as a list of referenced classes.

This user interface enables to replace pins representing folders by pins for class-level elements and sub-folders they contain, arranged in a circular layout. This feature allows users to navigate the hierarchical structure of the software system.

Relationships between Pinned Elements. If a pinned software element references another element also pinned on the same whiteboard, a curved line is drawn between their pins to show this connection ⓘ. These relation lines vary in thickness depending on how many source code references they represent, e.g., when showing relations between two folder pins.

Drawing Freehand Diagrams with Automated Conformance Checks. Users can pick up a virtual pen and draw freely on the whiteboards using a variety of colors ⓘ. Based on different drawing modes of the virtual pen, our tool distinguishes user's pen strokes into (1) uninterpreted drawing (for icons, text, etc.), (2) outlines around pins, and (3) arrows between outlines. Based on that, relation lines between pins are colored according to the freehand drawn arrows between outlines, providing users with a check on the conformance between their hand drawn arrows and the ground truth references in the subject system's source code [10].

Recording Audio. To capture elaborate thoughts, users can pick up a virtual microphone and create arbitrarily long audio recordings ⓙ. Once completed, they pin their audio recording to a whiteboard using the same gesture as for pinning software elements ⓚ.

Capturing VR Screenshots. To capture a specific view on the visualization, users can pick up a virtual camera and take VR screenshots ⓛ. These can then be pinned to a whiteboard as shown in ⓜ. When tipping on a VR screenshot pinned to a whiteboard, users can restore the visualization to the state of taking the picture, implementing a form of temporal snapshsot in the exploration process.

Reading Notes in Eclipse. To assist users in working in the source code based on the insights gained during VR explorations, they can access their whiteboard notes directly in Eclipse. For that purpose, ISA extends the Eclipse UI with a view enabling users to inspect whiteboards, zoom in and out, play audio recordings, and enlarge screenshots ⓝ. Further, users can open files of pinned elements in Eclipse by clicking on pins in the whiteboard view ⓞ.

2.2 Tool Architecture

Figure 2 provides a simplified overview of ISA's core components and their interconnections.

Model Server: Platform of Eclipse Plugins. ISA's model server is implemented as an extensible platform of Eclipse bundles. The lower part of Figure 2 provides an overview of that platform.

Automated Model Generation. ISA's overall model generation process is handled by a core plugin which successively executes a pipeline of three steps, each passing its results on to the next:

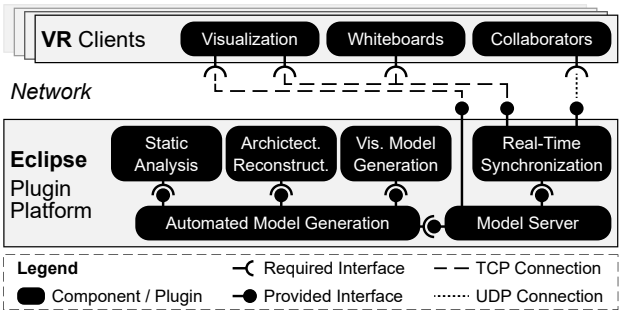

Figure 2: Architectural overview of ISA. A platform of Eclipse plugins provides an automated system analysis and a model server. VR clients connect to it and display a collaborative visualization.

(1) analyzing Eclipse projects selected by a user and storing the results in a model suitable for capturing the structure of an object-oriented software system from folder to member level,
(2) (optional step) employing an architecture recovery procedure [8] that replaces the folder-level structure in the results of Step 1,
(3) transforming the software model resulting from the previous step into a visualization model as input for VR clients.

A concrete solution for a step, e.g., an analysis for Java source code as Step 1, is implemented in form of an Eclipse bundle registering with the core plugin via an extension point. ISA's model generation platform can be extended with alternative solutions for individual steps, e.g., support for analyzing an additional programming language (Step 1) or an alternative mapping of software elements to visual elements (Step 3), while integrating with pre-existing solutions for other steps without further adaptions.

To further ease the extension of ISA's model generation platform, we define the structure of information passed between its steps via meta models, using the Eclipse Modeling Framework (EMF).

Communication via Network. To exchange information between Eclipse server and VR clients, ISA uses two channels:

1. UDP Channel. The ISA server establishes a dedicated UDP connection with each VR client for continuous sharing of position and rotation data for users' heads, hands, and interacted objects such as whiteboards, class cylinders, or cameras. This system does not implement additional measures to recover dropped network packets, as any lost data is overridden by the subsequent successful transmission's updated information.

2. TCP Channel. For information exchange that is less time critical but that requires reliable and ordered message delivery (e.g, showing relations via the relationship graph or closing folder sphere), the ISA model server provides an HTTP-based interface. Upon receiving events from a connected client, the server (1) verifies their consistency with a log of all pre-existing events, (2) persists a new version of the log with the inserted events, and (3) forwards the new events in their respective order to all connected clients.

We use the Eclipse Modeling Framework (EMF) to define meta models for the data structures in our TCP-based message exchange, to automatically generate equivalent code from the meta models in both Java (for use in Eclipse bundles) and C# (for use in the VR visualization client).

VR Visualization Clients. ISA's VR visualization client acts as local realization of the synchronized visualization state maintained by the ISA Eclipse server. That includes interactions carried out directly by the user of an ISA VR client – these first go through the model server and its verification before being sent back and then being applied in the order consistent with potential other events issued by collaborators in the meantime.

ISA's VR visualization client is based on the SteamVR platform and implemented in C# using the Unity 3D engine, making ISA's VR client compatible with all VR hardware supported by SteamVR.

3 CASE STUDY WITH PRACTITIONERS

We evaluated our tool in an exploratory case study with four software engineering practitioners. Working in pairs, participants used our tool to collaboratively explore an unfamiliar Java subject system. After these sessions, we used a questionnaire to collect participants' feedback on using VR for exploring an unfamiliar software system and taking notes on findings. Further, we analyzed the whiteboards created by both teams during their session and extracted all statements about the subject system they have noted. We then relayed these statements to the original developers of the subject system to assess their correctness and relevance in a re-engineering context. Below, we discuss general results of the study. More detailed documents and raw data is accessible in an online appendix[2].

Feedback from the subject system's original developers show that, while the relevance of participants' statements was mixed with some being vital for future work with the system's source code and others not concerning relevant aspects at all, the correctness of participants' statements was very high. Results from the analysis of the VR sessions and post-questionnaire show that ISA provides good support for an exploration on the level of architectural elements (folders and classes). Regarding note taking, participants valued the flexible nature of using freehand scribbling, recording audio, and taking screenshots. At the same time, they pointed out that it requires practice to fully utilize the immersive VR tool, especially for handwriting on the VR whiteboards. Further, they mentioned potential of an automated audio recording transcription feature.

4 CONCLUSION AND FUTURE WORK

We presented the collaborative VR software exploration tool Immersive Software Archaeology (ISA). We described its usage from a user's points of view and provided an overview of its architecture. Further, we reported on results from an exploratory case study with four software engineering practitioners.

In future work, we plan to extend ISA with support for automated audio-to-text transcription. Further, we plan to conduct quantitative studies involving larger samples of software engineering practitioners to investigate the tool's effectiveness, e.g., comparing ISA with traditional software exploration environments such as an IDE.

ACKNOWLEDGMENTS

Hoff and Seidl are supported by the DFF (Independent Research Fund Denmark) project "Immersive Software Archaeology (ISA)" (0136-00070B). Lanza is supported by the Swiss National Science Foundation (SNSF) project "INSTINCT" (Project No. 190113).

[2] https://doi.org./10.6084/m9.figshare.24499726

REFERENCES

[1] G Ann Campbell. 2018. Cognitive complexity: An overview and evaluation. In *Proceedings of the 2018 international conference on technical debt*. 57–58.

[2] Stephan Diehl. 2007. *Software visualization: visualizing the structure, behaviour, and evolution of software*. Springer Science & Business Media.

[3] S. Ducasse and D. Pollet. 2009. Software Architecture Reconstruction: A Process-Oriented Taxonomy. *IEEE Transactions on Software Engineering* 35, 4 (July 2009), 573–591. https://doi.org/10.1109/TSE.2009.19

[4] Florian Fittkau, Alexander Krause, and Wilhelm Hasselbring. 2015. Exploring software cities in virtual reality. In *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*. IEEE, Bremen, Germany, 130–134. https://doi.org/10.1109/VISSOFT.2015.7332423

[5] Dussan Freire-Pozo, Kevin Cespedes-Arancibia, Leonel Merino, Alison Fernandez-Blanco, Andres Neyem, and Juan Pablo Sandoval Alcocer. 2023. DGT-AVisualizing Code Dependencies in AR. In *2023 Working Conference on Software Visualization (VISSOFT)*. IEEE.

[6] Denis Gračanin, Krešimir Matković, and Mohamed Eltoweissy. 2005. Software visualization. *Innovations in Systems and Software Engineering* 1, 2 (Sept. 2005), 221–230. https://doi.org/10.1007/s11334-005-0019-8

[7] O. Greevy, M. Lanza, and C. Wysseier. 2005. Visualizing Feature Interaction in 3-D. In *3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*. IEEE, Budapest, Hungary, 1–6. https://doi.org/10.1109/VISSOF.2005.1684317

[8] Adrian Hoff, Lea Gerling, and Christoph Seidl. 2022. Utilizing Software Architecture Recovery to Explore Large-Scale Software Systems in Virtual Reality. In *2022 Working Conference on Software Visualization (VISSOFT)*. IEEE, Limassol, Cyprus, 119–130. https://doi.org/10.1109/VISSOFT55257.2022.00020

[9] Adrian Hoff, Michael Nieke, and Christoph Seidl. 2021. Towards immersive software archaeology: regaining legacy systems' design knowledge via interactive exploration in virtual reality. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, Athens Greece, 1455–1458. https://doi.org/10.1145/3468264.3473128

[10] Adrian Hoff, Christoph Seidl, Mircea Lungu, and Michele Lanza. 2023. Preparing Software Re-Engineering via Freehand Sketches in Virtual Reality. In *Proceedings of the 39th IEEE International Conference on Software Maintenance and Evolution*. IEEE. https://www.inf.usi.ch/lanza/Downloads/Hoff2023b.pdf

[11] C. Knight and M. Munro. 2000. Virtual but visible software. In *2000 IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*. IEEE Comput. Soc, London, UK, 198–205. https://doi.org/10.1109/IV.2000.859756

[12] Rainer Koschke and Marcel Steinbeck. 2021. SEE Your Clones With Your Teammates. In *2021 IEEE 15th International Workshop on Software Clones (IWSC)*. IEEE, Luxembourg, 15–21. https://doi.org/10.1109/IWSC53727.2021.00009

[13] Alexander Krause-Glau, Marcel Bader, and Wilhelm Hasselbring. 2022. *Collaborative Software Visualization for Program Comprehension*. https://doi.org/10.1109/VISSOFT55257.2022.00016 Pages: 86.

[14] M. Lanza and S. Ducasse. 2003. Polymetric views - A lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering* 29, 9 (Sept. 2003), 782–795. https://doi.org/10.1109/TSE.2003.1232284

[15] Mircea Lungu, Michele Lanza, and Oscar Nierstrasz. 2014. Evolutionary and collaborative software architecture recovery with Softwarenaut. *Science of Computer Programming* 79 (Jan. 2014), 204–223. https://doi.org/10.1016/j.scico.2012.04.007

[16] Leonel Merino, Mohammad Ghafari, Craig Anslow, and Oscar Nierstrasz. 2017. CityVR: Gameful Software Visualization. (2017), 5.

[17] Roberto Minelli and Michele Lanza. 2013. SAMOA – A Visual Software Analytics Platform for Mobile Applications. In *2013 IEEE International Conference on Software Maintenance*. 476–479. https://doi.org/10.1109/ICSM.2013.76 ISSN: 1063-6773.

[18] David Moreno-Lumbreras, Jesus M Gonzalez-Barahona, and Andrea Villaverde. 2021. BabiaXR: Virtual Reality software data visualizations for the Web. (2021).

[19] Harry Sneed and Chris Verhoef. 2019. Re-implementing a legacy system. *Journal of Systems and Software* 155 (Sept. 2019), 162–184. https://doi.org/10.1016/j.jss.2019.05.012

[20] Frank Steinbrückner and Claus Lewerentz. 2013. Understanding software evolution with software cities. *Information Visualization* 12, 2 (April 2013), 200–216. https://doi.org/10.1177/1473871612438785

[21] Richard Wettel and Michele Lanza. 2008. CodeCity: 3D visualization of large-scale software. In *Companion of the 13th international conference on Software engineering - ICSE Companion '08*. ACM Press, Leipzig, Germany, 921. https://doi.org/10.1145/1370175.1370188

[22] Richard Wettel, Michele Lanza, and Romain Robbes. 2011. Software systems as cities: a controlled experiment. In *Proceeding of the 33rd international conference on Software engineering - ICSE '11*. ACM Press, Waikiki, Honolulu, HI, USA, 551. https://doi.org/10.1145/1985793.1985868

[23] Sandra Yin and Julia Mccreary. 1992. Myths and realities: Defining re-engineering for a large organization. In *NASA. Goddard Space Flight Center, Proceedings of the Seventeenth Annual Software Engineering Workshop*.

[24] P. Young and M. Munro. 1998. Visualising software in virtual reality. In *Proceedings. 6th International Workshop on Program Comprehension. IWPC'98 (Cat. No.98TB100242)*. IEEE Comput. Soc, Ischia, Italy, 19–26. https://doi.org/10.1109/WPC.1998.693276