Comparing the EvoStreets Visualization Technique in Two- and Three-Dimensional Environments A Controlled Experiment

Marcel Steinbeck
University of Bremen, Germany
marcel@informatik.uni-bremen.de

Rainer Koschke
University of Bremen, Germany
koschke@uni-bremen.de

Marc O. Rüdel
University of Bremen, Germany
mor@uni-bremen.de

***Abstract*—Analyzing and maintaining large software systems is a challenging task due to the sheer amount of information contained therein. To overcome this problem, Steinbrückner developed a visualization technique named EvoStreets. Utilizing the city metaphor, EvoStreets are well suited to visualize the hierarchical structure of a software as well as hotspots regarding certain aspects. Early implementations of this approach use three-dimensional rendering on regular two-dimensional displays. Recently, though, researchers have begun to visualize EvoStreets in virtual reality using head-mounted displays, claiming that this environment enhances user experience. Yet, there is little research on comparing the differences of EvoStreets visualized in virtual reality with EvoStreets visualized in conventional environments.**

This paper presents a controlled experiment, involving 34 participants, in which we compared the EvoStreet visualization technique in different environments, namely, orthographic projection with keyboard and mouse, 2.5D projection with keyboard and mouse, and virtual reality with head-mounted displays and hand-held controllers. Using these environments, the participants had to analyze existing Java systems regarding software clones. According to our results, it cannot be assumed that: 1) the orthographic environment takes less time to find an answer, 2) the 2.5D and virtual reality environments provide better results regarding the correctness of edge-related tasks compared to the orthographic environment, and 3) the performance regarding time and correctness differs between the 2.5D and virtual reality environments.

I. Introduction

Software visualization exploits the human visual perception, in particular the ability to recognize patterns, to depict different aspects of a software system. In the last decades, various visualization techniques have been developed that are supposed to assist developers in assessing software regarding certain characteristics. Among these are *EvoStreets* proposed by Steinbrückner [1], which build on the software-as-a-city metaphor originally introduced as *CodeCity* by Wettel et al. [2], [3], [4]. EvoStreets show the hierarchical structure of a system (e.g., the package structure in Java) in terms of nested streets. The streets fork orthogonally at each level change in the hierarchy and streets corresponding to higher levels are wider than those corresponding to lower levels. Leaves of the hierarchy (e.g., classes in Java, depending on the visualization depth) are shown as cuboids in 3D, where the height, breadth, width, and color range depict certain metrics about a leaf in the hierarchy (cf. Figure 1). Generally, dependencies between nodes are often visualized by hierarchically bundled edges [5].

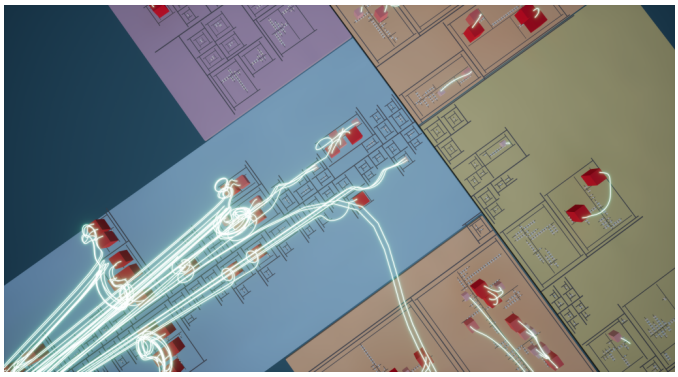


Fig. 1: Visualizing software clones with EvoStreets.

Initially, implementations of the CodeCity and EvoStreets approaches have used two- and three-dimensional rendering on regular two-dimensional displays, that is, they are rendered in isometric or 2.5D projection. When visualized from above, for instance, to give a birds-eye view, they may also be rendered in orthographic projection, loosing the third dimension (where a CodeCity essentially morphs into a classical Treemap [6]). Nowadays, though, immersive virtual reality systems (IVRS) as well as head-mounted displays (HMD) are available for the consumer market, allowing to bring the city metaphor into the virtual reality (VR). Studies have shown that HMDs may enhance the orientation in three-dimensional VR environments as they allow to rotate and move more naturally as opposed to traditional approaches [7]. Under the assumption that VR may also assists in the use of EvoStreets, by offering more intuitive interactions, tools supporting IVRS and HMDs have been developed lately [8], [9]. Yet, little research has been done on the benefits of using VR over using non-VR environments in EvoStreets.

Contributions. In this paper we present a controlled experiment in which 34 participants, by means of the EvoStreets approach, analyzed existing Java systems regarding software clones. After short introductions, each participant had to solve three different tasks in three different environments: Orthographic projection, 2.5D projection, and VR with HMD—one

task per environment. The experimental design was chosen to investigate the following research questions:

RQ1 Does orthographic projection allow a human beholder to solve tasks involving visually comparing the number of nodes or edges more quickly and accurately? 2.5D and VR may suffer from occlusion. On the other hand, 2.5D and VR offer one more dimension over orthographic projection, which can be exploited to change the perspective in order to inspect otherwise occluded elements.

RQ2 The main differences between 2.5D and VR are the immersion and the interaction. Does these differences have any effect on time and correctness between 2.5D and VR in task solving?

RQ3 The 2D orthographic environment offers only a two-dimensional projection. 2.5D and VR, on the other side, offer both three dimensions, but they differ in how one moves (keyboard versus controllers and head turning). Do these different interaction methods effect how humans move around?

The remainder of the paper is organized as follows. The next section presents related research. Section III describes our experimental design and Section IV discusses its results. Section V concludes.

II. Related Works

This section describes related research. We will first describe in more details the evolution of visualizations based on the city metaphor, which is the type of visualization in the focus of our experiment, and then summarize related work in the area of navigation and interaction design in virtual environments in general.

A. Code-City Visualization

Treemaps [6] are an early approach developed to depict the hierarchy of an arbitrary information structure in a space-economically manner, which are also well suited for visualizing the hierarchical structure of large size software systems [10] as well as to highlight hotspots regarding specific aspects [11]. The hierarchy is shown by recursively subdividing a given area, for instance, a rectangle, into subareas according to the structure that is to be visualized—classes in packages, files in directories, modules in subsystems, and so forth. This creates a tree of areas whose leaves represent particular software elements such as classes or files. The size of the leaves is proportional to a predefined software metric, for instance, lines of code, allowing to determine large elements quickly. Hotspots, on this basis, are depicted by coloring the leaf areas according to a second metric, for example, by applying a color gradient from green to red with respect to the cyclomatic complexity of an element.

Utilizing the third dimension, an additional metric can be depicted by mapping its value to the height of the corresponding surface, yielding three-dimensional blocks rather than two-dimensional areas [2], [3], [4]. Due to the effect of perceiving these blocks as a city, this technique is known as *CodeCity*. Treemaps and CodeCities, on one hand, are well suited to visualize large software systems in limited space but, on the other hand, have the disadvantage of being inconvenient when the underlying structure changes over time [12]. For example, when analyzing the evolution of a system—elements are added, removed, or relocated, the layout may change radically and thus the mental map of the observer may be lost. Moreover, comprehending the hierarchy is a challenging task due to missing distinct patterns in the city.

To overcome these shortcomings, Steinbrückner introduced a visualization technique named *EvoStreets* [1]. Harnessing the city metaphor, EvoStreets visualize a software's structures using road junctions rather than subdivided areas. That is, each level of the hierarchy is depicted by an individual line, representing a street within the city. The lower a level is, the thinner the corresponding street gets. The leaves of the hierarchy, in turn, are depicted using three dimensional blocks as in the CodeCity approach. Subsequently, the generated layout allocates more space but, at the same time, is more robust to changes in the underlying structure and may feature more distinct patterns of city districts that help in navigation thereby maintaining the mental map.

Our long-term goal is to visualize evolving software. For this reason, we have chosen EvoStreets for our experiment because—in contrast to CodeCities in the line of Wettel's seminal work [13]—it has a strong continuity over changes between versions and provides a structure that stretches itself out, leaving more space between objects of interest [1]. Moreover, in early prototypes by which we explored CodeCities in VR we made the experience that everything looks alike if one walks on the street, similarly to a stroll through Manhattan with skyscrapers left and right. Likewise, if one flies above the city to get an overview, the city appears always as nested rectangle which offers little visual cues for preserving the mental map. Because the layout of EvoStreets is much more ramified and its branches often have very different lengths and patterns, it is easier to turn them without loosing the orientation. And last but not least, for virtual environments that are meant to be explored and not only to be seen from afar we can take advantage of the infinite virtual space in VR when rendering EvoStreets.

Knight and Munro gave an overview on VR for software visualization as early as 2000 [14]. Since then various other researchers have used VR and 3D for software visualization for static information [15], [16], [17], [18], [19], [20], [21], [22] or for dynamic data such as resource bottlenecks or memory leaks [23], [24]. Elliott et al. discuss the affordances and challenges VR in software engineering and present ideas on how it can be used for code reviews [25]. Baum et al. developed a tool for prototyping and evaluating different kinds of software visualizations in 3D [26]. We refer the reader interested in software visualization in general to one of the many surveys that exist on this subject [27], [28], [29], [30],

[31], [32], [33], [34], [35].

B. Virtual Reality Versus Desktop

Navigation in virtual environments has been researched for a long time outside the software visualization community [36], [37], [38] and will continue to be a research topic because of the fast progress in hardware and rendering techniques. Sousa Santos et al. give an overview on the virtual environments, discussing several papers that compared HMDs to desktop environments [39]. In their experiment on navigation in a virtual maze, they found that—although users were generally satisfied with the VR system and found the HMD interaction intuitive and natural—most performed actually better in the desktop environment.

Ruddle et al. investigated the navigation in computer-simulated worlds with HMDs and with traditional desktop environments [40], [41]. In one of their experiments, participants had to navigate within large virtual buildings, consisting of one floor with nine rooms and one corridor [40]. The experiment was a repeated round tour through several rooms. In a second experiment, they looked into proprioceptive feedback and its influence on navigation within a virtual maze [41]. In the first experiment, they found that the HMD had an advantage to the speed of the participants [40], in contrast to their later experiment [41] and to Sousa Santos et al.'s experiment [39], which both took place in a maze. What exactly caused these conflicting results is a subject of further research.

Other experiments targeted at spatial cognitive capabilities within virtual environments. *Homing tasks* [42]—also known as *triangle completion tasks* [43]—were used to observe the influence of the amount of visual-only spatial information on the spatial orientation of humans. The task of the experimental subjects was to move a cylinder in a very constrained virtual environment. The researchers found a strong association of triangle layout on homing performance, but no effect of geometrical fields of view; that is, variations in the amount of simultaneously visible spatial information did not influence the acquisition of spatial knowledge in the environments used. Whether and how these results obtained for very specific geometrical problems can be transferred to navigation in virtual software cities is unclear.

Compared to classical software visualizations, 3D representations of software have the added value of the spatial dimension. However, in our scenario, this spatial dimension can only be used by explorers of the software if they have orientation. In order to acquire orientation, it is necessary to move in the environment [37], [44]. *Locomotion* comprises ways to move through virtual worlds such as teleportation, flying, etc. Variants of locomotion differ also in who is in control. A system can transport a user from one point to another one, or a user can decide on his or her own where to move. Bowman et al. [36] investigated the impact of different types of locomotion on the spatial orientation in virtual environments in a maze-based experiment. They found that flying controlled by the human better supports spatial orientation. Moreover, continuous movement is advantageous for the formation of a spatial model [36], [37], [45], [44].

In our experiment described later, we chose user-controlled flying with a constant speed. This way of locomotion has shown to be simple and intuitive and decreases the chances of sickness. A flight metaphor offers—even in reality—the highest flexibility of movement, continuous updating of the spatial perception (as opposed to teleportation), and a bird's eye view on demand. Many other types of movement sometimes used in modern computer games seem to be of limited suitability for the acquisition of spatial orientation because they tend to lead to disorientation [36].

Many of these experiments described above have been conducted in small and/or maze-like virtual environments not showing software data. So for the purpose of software comprehension through exploration of code-city visualizations findings might differ. This is especially true for visualizations of modern software systems that consist of several thousands of entities shown as buildings. Very recently, however, Rüdel et al. have conducted an experiment on how successfully humans orient themselves in EvoStreets in 2.5D and VR, respectively [9]. The participants had to solve a homing task, that is, find their way back to the place where they started. Their primary hypothesis was that the HMD environment would make it easier to gain a spatial orientation inside of software cities in comparison to a 2.5D environment but they found no statistically significant evidence for it in their controlled experiment with 20 participants

In an experiment, interaction must be easy to learn, so that neither the interaction nor learning effects influence the results too much. Mine provides an overview on different approaches for 3D and VR [46] as well as Guan and Zheng, and Nielsen et al. in regards to gestures [47], [48]. In our experiment we decided to use the simple pointing gesture for the HMD to move in our EvoStreets, using standard VR controllers activated with a button on the controller. For the desktop environments rendering the orthographic projection and 2.5D, we used a mouse and keyboard based on the common WASDQE input method widely used in computer games and applications for graphic modeling (the letters in WASDQE are the keys on the keyboard to be pressed for movements). Other research that compared the WASDQE input method were described by McMahan et al. [49], Nacke et al. [50] and Rüdel et al. [9].

III. Controlled Experiment

In this section, we describe our controlled experiment addressing our research questions outlined in Section I. We will first introduce the three different environments compared in the experiment (orthographic projection, 2.5D, and VR), then state our operational hypotheses derived from our research questions, describe the experimental subjects and objects, and what kinds of software attributes were mapped onto which visual features of the EvoStreets, and finally the tasks to be solved by the participants.

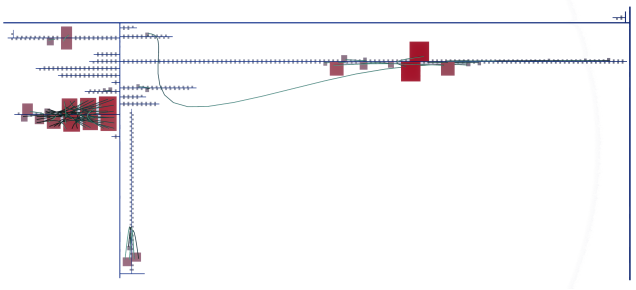


Fig. 2: The EvoStreets of Guava used as tutorial for the 2D environment.

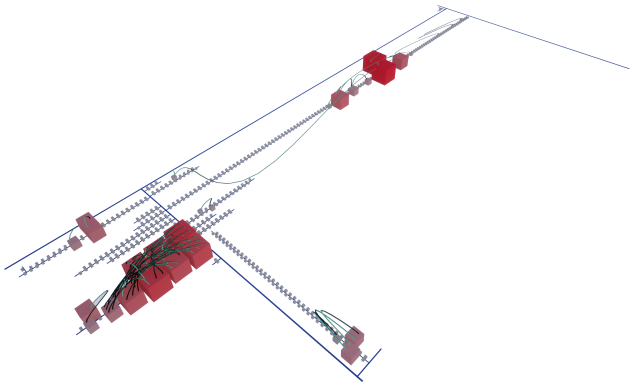


Fig. 3: The EvoStreets of Guava using the same visual attributes as in Figure 2, but rendered in three dimensions.

The tasks were derived from a typical visual-analytics context in which a human beholder must investigate cloning in a software system by visually compare nodes and edges in EvoStreets. For details refer to Section III-D

A. Compared Environments

The main objective of our experiment was to investigate the use of EvoStreets in the environments described below. Accordingly, each task to be solved by the participants took place in one of these environments.

2D orthographic environment (2D): The EvoStreets of the corresponding task are rendered in orthographic bird's eye view on a regular two-dimensional display as shown in Figure 2. Hence, only a block's area viewed from the top, but not its height is perceivable. Using a keyboard and mouse, the participants were able to move, rotate, and zoom the scene as desired.

2.5D environment (2.5D): The EvoStreets were projected to the display similar to the orthographic environment but rendered in three dimensions as shown in Figure 3. By adjusting the scene—again using a keyboard and mouse—the participants were able to move within the city and view it from different perspectives.

Virtual reality environment (VR): This environment uses the same three-dimensional rendering approach as the 2.5D environment, but presents the EvoStreet of the corresponding task on a stereoscopic head-mounted display. Unlike the other environments, this allows to view the entire city in a real three-dimensional, stereoscopic manner. By rotating their heads and bodies, as well as by using hand-held controllers, the participants were able to adjust the scene as needed. For our experiment, we used the head-mounted display and hand-held controller kit from HTC Vive.

We used the EvoStreets implementation developed by Rüdel et al. [9]—named *SCOOP*—who offered us their implementation as an Unreal Engine 4 plugin[1]. The plugin is capable of rendering EvoStreets for arbitrary software systems. Metrics can be depicted by mapping their values to the width, breadth, height, and color of the corresponding elements, respectively. Likewise, relations between elements can be visualized with hierarchically bundled edges which we used to show that two files are mutually cloned. All environments show exactly the same information so that a fair comparison can be made.

We only had to specify those visual parameters but otherwise did not have to make any changes in Rüdel et al.'s implementation except for one addition. We extended SCOOP by a position tracking component. During the task completion, this component continuously logged the location and the rotation of the participant, if necessary the orthographic distance of the environment and other tasks specific data.

For all tasks we used the same simple gaming-ready PC with an NVIDIA GeForce GTX 1070 graphics card and a standard $24''$ LCD display for the desktop tasks. More details on which software aspects are mapped onto which visual attributes follow in Section III-E.

B. Hypotheses

Research question RQ1 is asking whether there is any difference in human performance in tasks involving visually comparing the number of nodes or edges in EvoStreets for the three different environments (2D, 2.5D, VR). In contrast to 2D, utilizing the third dimension, 2.5D and VR are capable of depicting an additional metric by mapping the metric value to the height of the corresponding area. This, however, leads to a visualization in which nodes obscure each other in certain situations. Thus, additional movement interactions are required to view a city from different perspectives in order to access all available information. Compared to orthographic projection in 2D, these movement interactions engross more mental capacity and, thus, may have a negative effect on the time required to find an answer. Moreover, 2D, by design, is well suited to give a fast overview of the visualized scene, allowing to compare elements quickly. This leads us to our first hypothesis:

***H1.T*:** In EvoStreets, the time required to find an answer using orthographic projection in 2D is shorter than using 2.5D projection or virtual reality.

As mentioned before, 2.5D and VR offer more advanced movement interactions to view a city from different perspectives. This may in particular be useful when exploring a city

[1]https://www.unrealengine.com

with thousands of blocks and edges connecting blocks. In orthographic projection edges overlie blocks and, thus, may be more difficult to distinguish—the more edges are visualized the more visual clutter is present. This may have a negative effect on the correctness of findings. To test whether this is the fact, we formulate our next hypothesis:

***H1.C*:** Regarding edge-related tasks, there are more correct answers in EvoStreets using 2.5D projection or virtual reality than in EvoStreets using orthographic projection.

Note that *H1.C*refers explicitly only to edge-related tasks where the overlay of blocks and edges causes visual clutter. If a task requires to compare nodes rather than edges and the edges occlude nodes, edges could simply be hidden on demand by the user. Hiding nodes in edge-related tasks, however, does not work because otherwise the source and target of edges would disappear.

Hypotheses *H1.C* and *H1.T* are operationalizations of research question RQ1, in which we relate two-dimensional visualization (orthographic projection) versus three-dimensional visualization (2.5D and VR). In research question RQ2, we relate only the two three-dimensional visualizations, namely, 2.5D and VR, to each other.

There are reasons to believe that VR allows one to more closely inspect details, which should correspond to more correct results. VR offers a more immersive experience and the interaction is more natural: head turning immediately allows one to seize a block and compare it to others or to visually follow an edge from a given point of view. Humans are very good in focusing objects and using their hands (here: holding controllers) in parallel and independently, which is less true for looking at a fixed picture shown on a monitor and hitting keys on a keyboard for interaction. On the other hand, there are also reasons to believe why 2.5D may ease finding correct answers. Developers are familiar with computer monitors, mice, and keyboards. And viewing the streets from behind a windshield—similar to the impression of looking at EvoStreets in 2.5D—is a natural experience, too. 2.5D offers a "frozen" view which may help in comparing nodes and edges while the eye wanders around the scene. The immersion in VR may even distract the human beholder from the actual task. In summary, we have no sufficient arguments why either of the two environments is expected to be better than the other one, but we have enough reasons to believe that they may at least be different. For these reasons, we postulate the following undirected hypothesis:

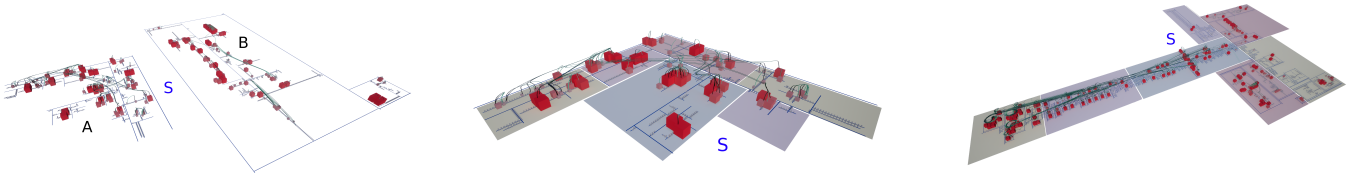
***H2.C*:** The correctness in solving tasks related to nodes as well as edges differs in VR and 2.5D.

Many developers are well familiar with navigation using a keyboard. They can navigate quickly in their integrated development environments using keyboard and mouse. Many developers also have a background in computer games, where WASDQE navigation is very common. They are expected to be less familiar with navigation in VR. Even if they use VR every now and then, overall they will spend their time mostly in front of a computer with keyboard and mouse. Hence, we might expect that they move faster in 2.5D than in VR. As a matter of fact, Rüdel et al. found in their experiment comparing 2.5D and VR for homing tasks that their participants moved somewhat faster in the 2.5D rather than VR environment, although the difference could not be shown to be statistically significant [9]. One most note though that navigation is only one part of the overall effort for task solving. Inspecting details is another one. Because VR may arguably offer better means to inspect details (see the discussion above for hypothesis *H1.C*), we expect our participants to also take advantage of these, which may require additional time or save time—we cannot know in advance. In summary, there are pros and cons for both environments and we have no strong conjecture that one outperforms the other one, yet we think there are enough reasons to assume that they are different as formulated by the next undirected hypothesis:

***H2.T*:** The time needed to solve node as well as edge related tasks differs in VR and 2.5D.

The last research question RQ3 asks whether the use of 2.5D (with monitor, mouse and keyboard) or VR (with HMD and controllers) has any effect on how humans moving in these different environments. As a matter of fact, we did not formulate an operational hypothesis upfront to be falsified/confirmed in the experiment. Rather we recorded the trajectory of the participants and, much to our surprise, found a difference when we overlaid these after the experiment for 2.5D and VR. Because this observation arose in the course of the experiment and was not expected at its outset—and also because it is difficult to quantify, we refrain from formulating it as an operational hypothesis here. Rather we will tackle this research question more qualitatively.

We note that weaknesses of either one of the three different environments may be compensated by additional interaction. For instance, in orthographic projection one could simply hide the edges on demand by the user to have an unblocked view on all blocks. Yet, that requires additional interaction. EvoStreets are claimed to be designed to answer the kinds of questions asked in our experiment immediately and visually by simply looking. Likewise, all the answers to our questions could be computed by algorithms so that a human would not even have to look at the EvoStreets. However, EvoStreets are advocated as a tool for visual analytics that leverages a human's immediate visual perception. Visual analytics acknowledges that many questions are raised only through looking at a visualization. It does not assume that a human analyst is aware of all possible questions in advance, even more so a developer of a tool intended to support software analytics cannot know in advance all possible questions a user of her/his tool is trying to answer later. Visualization is intended to be helpful at an early stage of an investigation where concrete hypotheses cannot be formulated yet, let alone precise answers be computed.



(a) Task 1: Which of these two systems contains more fragments cloned in other source files?

(b) Task 2: Which pair of subsystems is mutually cloned the most?

(c) Task 3: Which subsystem accommodates the majority of source files containing clones?

Fig. 4: Visualization of the scenarios of our three tasks as 2.5D projections. *S* marks the starting point of the exploration.

C. Participants and Experimental Objects

In order to verify our hypotheses, we conducted a controlled experiment with 34 participants. They were recruited through convenience sampling. The sample comprises 17 advanced undergraduate students of a running VR project on EvoStreets in our research group, 5 students of software engineering courses taught by our research group at graduate and advanced undergraduate levels, 3 other students, 5 researchers of our research group, 2 scientific assistants from different research groups at our university, and 2 professional developers. Among these were 3 women. Their age ranged from 19 to 63. 28 out of the 34 participants had prior experiences in VR. All of them are formally trained computer scientists (although not all of them have their formal degree yet as explained above). To counter learning effects between tasks, we randomly separated the participants into six different groups with 5 to 6 members, each in turn using the environments in latin squared order. Participation was voluntary. Neither any incentive nor disincentive was given for participation and there existed no time pressure.

As stated before, by means of the EvoStreets visualization approach deployed in different environments, each participant had to solve three different tasks in which existing Java systems had to be analyzed regarding software clones. We decided to use software clones because studies have shown that clones are a common issue in daily software development, therefore being relevant for all kinds of developers [51]. Moreover, a multitude of sophisticated clone detection tools are available, allowing to examine and export findings conveniently. Using the Axivion Bauhaus Suite[2], we analyzed the Java systems in Table I. Guava was used as a training system by the participants before each task until they felt comfortable with the environment. All participants began at the same starting point, denoted as *S* in Figure 4.

TABLE I: Java systems used for our experiment. Guava was used for training.

Name	Lines of Code	Files	Findings
(Guava)	75,042	516	53
Jillion	75,520	929	66
JRuby	227,145	1360	110
Spring Boot	181,795	3042	228

[2]https://www.axivion.com/en

D. Visual Mapping

Figure 2 shows the EvoStreets of Guava used for the tutorial in 2D and Figure 3 the same rendered in three dimensions. The street layout is auto-generated based on the hierarchy of the files and directories. Every street in our EvoStreets visualization represents a directory containing Java source files or other directories, the nodes represent Java source files and an undirected edge between two nodes indicates that these two Java source files contain similar code fragments, so called clones. In order to identify files that are mutually cloned, we used SCOOP's hierarchically bundled edges to visually connect the Java files sharing clone code [5].

As a metric, we used the clone rate (fraction of cloned tokens among all tokens) of each file. We mapped the metric value proportionally onto the area of the ground square (width and breadth are chosen equally such that their product is proportional to the clone rate) as well as onto height and color range (obviously, height is never visible in the orthographic projection). That is, the higher a file's clone rate is, the larger (in all three dimensions) and more deeply red-colored its block gets.

Mapping the clone rate equally to the size and color of a block is due to the circumstance that in the orthographic environment the height of a particular block is not perceivable, therefore being at a disadvantage if a second clone-related metric is expressed using the height. Moreover, the same information can be viewed from all angles, no matter if one looks from the top or walks between the buildings in the EvoStreets. This redundant mapping ensures that exactly the same information is available in all three environments at all times. In other words, none of the environment neither gave an advantage nor created extra noise, which could have confused participants.

E. Tasks

In the following, we will describe each task in more detail. The three tasks were derived from a list of user goals commonly associated with cloning as collected by Basit et al. [52]. The need for inspecting and comparing clones in maintenance tasks of systems with clones is very common among these goals [52]. Yet, the fact that clone information is shown in EvoStreets is not of utmost importance—the tasks can be generalized to all kinds of problems in which hierarchical graphs with numerical attributes are to be investigated.

Task 1: In the first task, the participants had to determine which of two systems contains more cloned fragments shared among different files. For that, we placed two different systems side-by-side in the same visualization, both of which contained clones but are not cloned with each other. The systems to be compared should be of similar and sufficient size and have similar cloning between files so that the task does not get trivial. We found the systems Jillion and JRuby fitting these requirements, denoted as (A) and (B) in Figure 4a. That is, the participants had to estimate and compare the number of non-recursive edges in both systems. This is particularly difficult in the orthographic environment, which does not support viewing overlapping edges from all three-dimensional perspectives.

Task 2: In the second task, the aim was to identify the two subsystems of a system that are mutually cloned the most, again by counting edges. For that, we subdivided a system into distinct subsystems of similar sizes using colored rectangles to emphasize each subsystem. The human visual perception cannot process too many colors at the same time, thus, a visualizer must often reuse the same color. As a realistic visual difficulty, we colored two of the subsystem in the same color. However, each subsystem was clearly identifiable by its rectangular shape and the similarly-colored subsystems were placed remote from each other. We used Jillion for this task. The resultant EvoStreets are shown in Figure 4b.

Task 3: In the final task, the participants had to locate the subsystem that accommodates the majority of the source files containing clones, by comparing the number of red-colored blocks. We choose Java Spring Boot as a good candidate. The corresponding EvoStreet is shown in Figure 4c. Similar to the second task, the system was subdivided into subsystems depicted by distinct subsystems.

In the further course we will speak of Tasks 1 and 2 as edge-related, while Task 3 is node-related.

IV. Results

For each of the three tasks described in Section III, we measured the time required to find an answer as well as its correctness. Furthermore, we gathered the trajectories of the participants during task completion. At the end of all tasks, each participant had the opportunity to rate the usefulness of each environment for solving the corresponding task. In the following, we present and discuss the results with respect to time, correctness, perceived usefulness, and trajectories.

A. Time (H1.T and H2.T)

Table II shows the minimum, maximum, mean, and median times that were required to find an answer for each task and environment. According to the mean times, one might suspect noticeable deviations in the reddish emphasized values. Yet, the median times show no peculiarities. As depicted by the boxplot in Figure 5, the differences between mean and median are due to several outliers: two outliers in Task 1/VR, one outlier in Task 2/2.5D, and two outliers in Task 3/VR.

The larger one of the two outliers in Task 1/VR as well as the outliers in Task 2/2.5D and Task 3/2D come from the same participant who, at that time, had any experience with neither software visualization nor VR. Whether the participant was riveted by the visualization, leading to an unusually high processing time or any other individual aspect is the cause, we cannot tell.

TABLE II: Time for task solving [in secs]

Task	1			2			3		
Environment	2D	2.5D	VR	2D	2.5D	VR	2D	2.5D	VR
min	20	69	56	51	32	56	69	32	54
max	331	205	953	152	335	208	304	381	362
mean	155.6	115.5	234.4	92.0	136.3	109.6	133.3	138.5	160.8
median	115.0	107.0	129.5	89.0	119.5	105.0	107.0	111.0	124.0

As can be seen, the participants, on average, were fastest in Task 2 and Task 3 when using the 2D environment, underpinning our hypothesis *H1.T* (the time required to find an answer in EvoStreets using 2D versus 2.5D and VR). Yet, 2.5D outperformed the other environments in Task 1 (depicted by the green-colored values). In order to validate the differences in *H1.T*, we ran the non-parameterized Kruskal-Wallis test on the median times of each task, comparing the groups 2D, 2.5D, and VR, respectively. Based on our results (with p-values for the null hypotheses as follows: Task 1: 0.7, Task 2: 0.3, and Task 3: 0.9), there is no significant difference among the three environments for all tasks; thus, we must reject *H1.T*.

In order to exclude learning effects, we examined the median times for each of the six environment permutations but neither found any statistically significant difference. So the order of the environments usage seems not to have any learning effect. Likewise, we did not find significant differences regarding the median times between the 17 students of our VR project and the remaining participants. Even though these 17 students had practical experience in EvoStreets and could have presented a potential bias, we can now safely exclude this aspect.

To validate hypothesis *H2.T*, that is, there is a significant difference regarding the time required to find an answer in 2.5D and VR, we ran the non-parameterized Mann-Whitney-Wilcoxon test. In contrast to our expectations, we could

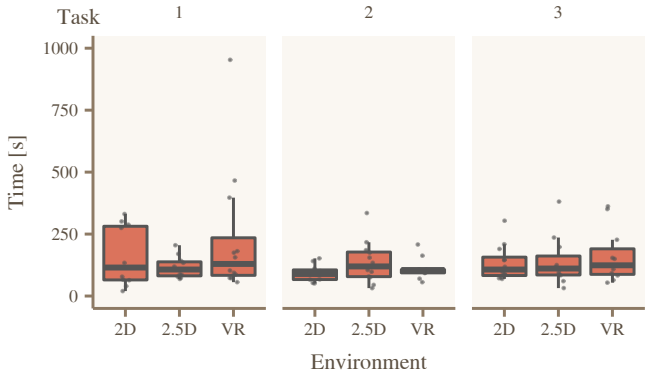


Fig. 5: Time required to find an answer of each task and environment.

not find a statistically significant difference (p-values are as follows: Task 1: 0.5, Task 2: 0.6, Task 3: 0.7). Thus, we must reject *H2.T*.

B. Correctness (H1.C and H2.C)

Table III shows the number of correct and incorrect answers for each task and environment. The percentages of correct answers in Task 1 (depicted by the last row of the table) are relatively similar. In contrast, there are noticeable differences in Task 2 and Task 3. Using 2D and 2.5D, almost all participants solved Task 2 correctly. On the other hand, in VR only six out of eleven participants found the correct pair of subsystems.

TABLE III: Number of correct and incorrect answers of each task and environment (the groups are not of equal size).

Task	1			2			3		
Environment	2D	2.5D	VR	2D	2.5D	VR	2D	2.5D	VR
correct c	8	7	7	10	11	6	5	1	3
incorrect i	3	4	5	1	1	5	7	10	8
total $c+i$	11	11	12	11	12	11	12	11	11
c/(c+i)	0.72	0.63	0.58	0.90	0.91	0.54	0.41	0.09	0.27

Hypothesis *H1.C* relates explicitly only to edge-related tasks (see Section III-B). In both Task 1 and Task 2, participants had to look out for edges (but not in Task 3). The imbalance of the results of these two tasks regarding the VR environment may suggest that Task 2, although being similar to Task 1, has a specific feature which puts VR at a disadvantage compared to 2D and 2.5D. For Task 1, two given systems had to be compared, whereas Task 2 required to compare 15 different pairs of subsystems. Whether that may serve as a hint, we can only speculate. At any rate, in both tasks, VR has yielded the worst results. Using Fisher's test, we checked the difference in correctness pairwise. The resulting p-values did not reach a sufficient significance level for rejecting the null hypothesis (T1: $p_{2D/2.5D}$: 1.0 / $p_{2D/VR}$: 0.67, T2: $p_{2D/2.5D}$: 1.0 / $p_{2D/VR}$: 0.15), so we must reject hypothesis *H1.C*, which claims that, regarding edge-related tasks, there are differently many correct answers in EvoStreets using 2.5D or VR versus using 2D.

Unlike Task 1 and Task 2, Task 3 focuses on comparing red-colored blocks rather than edges. Thus, the results of Task 3 in Table III may be an indication that comparing blocks in 2.5D is more error prone than in 2D or VR. Even more to our surprise, 2D yielded the best rate of correctness for Task 3 although we assumed that occlusion of blocks through edges may make it harder to correctly compare the number of blocks. However, the differences are neither statistically significant: (Task 3: $p_{2D/2.5D}$: 0.16 / $p_{2D/VR}$: 0.67), hence, we cannot assume that 2D is truly better than either 2.5D or VR.

As already for the median times, we compared the answers of the 17 students of the VR project with the answers of the remaining participants regarding correctness to exclude a potential bias. Again we did not find any significant difference, hence, can exclude the bias.

Similar to *H1.C*, we ran Fisher's exact test to validate hypothesis *H2.C*, which claims that there is a significant difference regarding the correctness of answers given in 2.5D and VR. No statistically significant difference could be found, though (p-values are as follows: Task 1: 1.0, Task 2: 0.07, Task 3: 0.6). Accordingly, we cannot accept *H2.C*.

C. Ratings

At the end of all tasks, each participant had the opportunity to rate the usefulness of each environment from 1 to 6, with 1 being the best rate and 6 the worst. Figure 6 shows the accumulated results. The averages were as follows: 2D: 3, 2.5D: 2, VR: 2. Overall, the two three-dimensional environments were rated better than 2D. Interestingly, the subjective assessment is not well reflected in the objective measures on timing and correctness. The participants may like 3D visualization, but they do not seem to offer any real advantage for task solving over 2D.

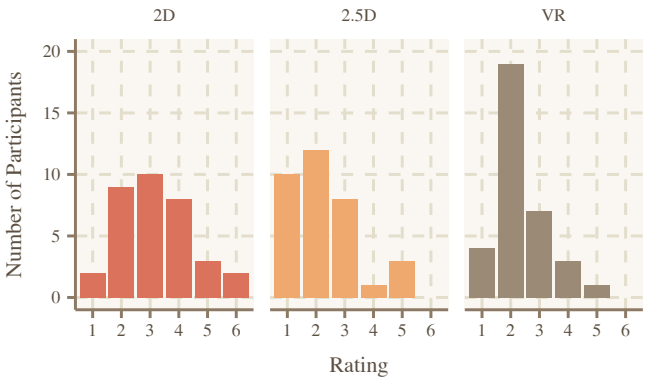
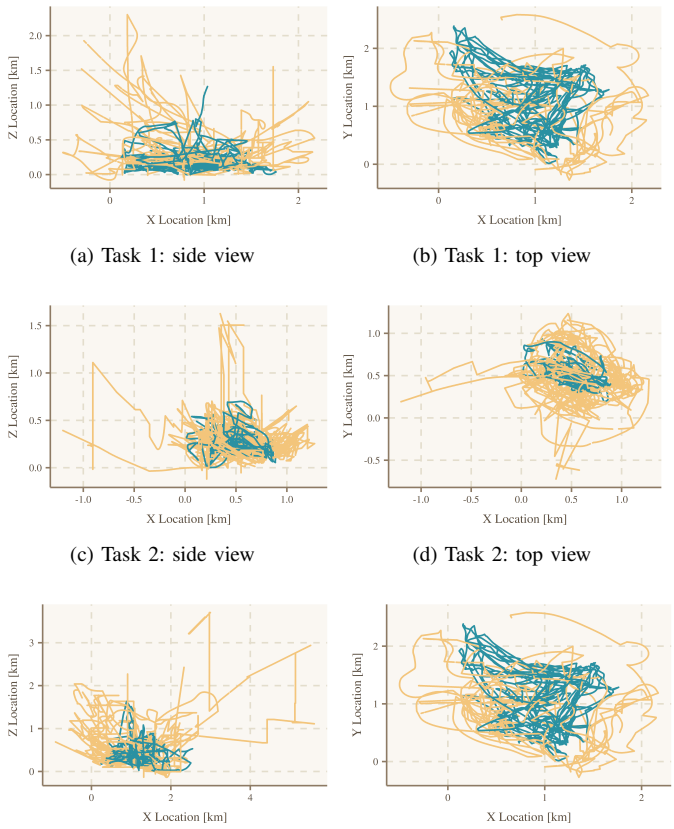


Fig. 6: Rating of each task and environment.

D. Trajectories

While the participants have solved the tasks in 2.5D and VR, we gathered their position data within the map to find differences in movement patterns. Figure 7 shows the resulting trajectories for each of the three tasks in the side view as well as in the top view. The yellow-colored paths depict all position data of the 2.5D environment whereas the cyan-colored paths depict all position data of the VR environment. A visual pre-analysis suggests that the movement patterns differ. That is, movements in the 2.5D environment seem to be more extensive compared to movements in the VR environment. To express the visual differences in numbers, we calculated two metrics for each set of position data (that is, the position data of a participant): 1) *Path Length*, which accumulates the distances between each two consecutive movement points and 2) *Average Speed*, which puts the length of a path in relation to the time that was required to walk this path. Furthermore, we determined the unweighted *Range of Movement* metric for each task and environment by first calculating the centroid (unweighted mean of the recorded movement point locations) and then calculating the maximum of the distances of all movement points to its centroid. This number gives an idea on the "volume" space of trajectories.



(a) Task 1: side view (b) Task 1: top view

(c) Task 2: side view (d) Task 2: top view

(e) Task 3: side view (f) Task 3: top view

Fig. 7: Trajectories of the 2.5D (yellow) and VR (cyan) environments.

We found differences in the median values of all three aspects between 2.5D and VR for almost all tasks. Some of which were significant (cf. Table IV). We have noticed this difference in movement patterns as a byproduct of our experiment. We have not anticipated it, hence, have neither controlled it or took any measure in advance to find explanations. We plan to run a study in the future to further delve into this phenomenon, but found it to be interesting enough to share it already now with other researchers.

TABLE IV: P-values for the Mann-Whitney-Wilcoxon tests of the three inspected aspects of the trajectories in 2.5D vs. VR.

Task	1	2	3
Path Length	0.30	0.0040	0.10
Average Speed	0.20	0.0010	0.01
Range of Movement	0.09	0.0002	0.04

E. Threats to Validity

This sections discusses potential threats to validity of our experiment.

Internal validity: To make sure that any observed effect in our dependent variables can solely be attributed to the independent variable, we took the following measures. To counter influences of prior knowledge and experiences of the participants, we asked each participant to solve tasks in every environment (orthographic projection, 2.5D, VR) and otherwise assigned them randomly to one of the six orders of experimental steps. To minimize sequence effects, for instance, through learning, we shuffled the tasks and environments into six different orders. Differences in prior experiences in VR may influence the participants' performance. Yet, almost all participants (28 out of 34) had prior experiences in VR and all were trained before the experiment in each of the three different environments. The training sessions did not have any time limit. The participants were offered all available time to get to know the environment and to feel comfortable with it. Yet, we cannot fully exclude that the "fun factor" of VR, which may have caused them to spend more time than actually needed to solve the tasks in VR during the actual experimental run, in particular, because we told them before each task that there is no time pressure. They should use as much time as they feel is needed to find the answer.

To limit the observer-expectancy effect—the form of reactivity in which a researcher's cognitive bias causes him or her to subconsciously influence the participants of an experiment—we used quantitative measurements of the dependent variables: time and number of correct answers, which can both be objectively assessed. Yet, there could have been an subconscious influence during training. The likelihood of the subject-expectancy effect—a form of reactivity when a participant expects a given result and therefore unconsciously affects the outcome—may be higher than the observer-expectancy effect because 17 participants were students of a VR project in our group and 5 participants were researchers of our group (although from working areas different from visualization). Although we did not tell any participant in advance the goal of our study, we cannot fully exclude that the subject-expectancy effect may have taken place.

Because we do not know the distribution of the population from which we drew our participants and because the number of participants would not justify that the central limit theorem applies, we used non-parameterized statistical tests. Because outliers have a larger influence in small samples, we used statistical tests that treat the quantitative measures only as ordinal data. The downside of these ordinal tests is that they do not take into account the actual time differences. They only rank the times. We also ran the parameterized and interval-scaled ANOVA and t-tests, even though their preconditions cannot be assumed, to get an estimate as to whether the actual differences rather than just the rank of the time give a different picture. Yet, even then no significant differences could be found.

External validity: Here we discuss how far our results are generalizable, given that the experiment was conducted on a sample of the relevant population and in a controlled environment.

We used cloning information for three Java systems to produce the visualization. Other types of relations, metrics, and hierarchies in software may yield other types of EvoStreets.

In particular, the size of the systems and the number of edges is expected to have an influence. The sizes of the programs visualized as EvoStreets in our experiment were in the range from about 75 KLOC to 227 KLOC. Smaller and larger programs may lead to other results.

The majority of participants had prior experience in VR. That is, we cannot claim that the results of our study can be generalized to developers without prior VR experience. Yet, the required level of experience can be gathered quickly by some training. One needs to always learn to "read" a visualization to make optimal use of it. Controlled experiments are generally somewhat artificial. Only through long-term studies in the field—thus within software development organizations visualizing their own software, a visualization as well as any other software engineering tool or method can be truly assessed.

We used convenience sampling to recruit our participants because it is generally very difficult to employ any other more reliable sampling strategies due to lack of participants and prior knowledge on characteristics of the population. Convenience sampling, however, bears the risk that the sample does not reflect the population well.

Construct validity: Construct validity is the degree to which a test measures what it claims, or purports, to be measuring [53]. In our experiment, construct validity is reached when the experimental treatment reflects the construct of cause of our theoretical model and when the outcome reflects the effect truly. We claim that the use of EvoStreets for cloning as well as the given tasks are realistic treatments and time and correctness are appropriate measures of the effect. We derived those tasks from a taxonomy of clone-related goals and information needs of the literature [52]. Time and correctness are meaningful measures of performance.

V. Conclusion

The EvoStreets visualization is a well suited approach to visualize the hierarchical structure of a software system and to depict hotspots therein. Thanks to recent technological improvements, EvoStreets can now be explored in virtual reality. Yet, there is little research on comparing EvoStreets using virtual reality systems with EvoStreets using conventional environments. To gain further insights, we conducted a controlled experiment in which 34 participants, by means of the EvoStreets visualization technique, had to analyze cloning in existing Java systems. The experiment was subdivided into three distinct tasks, each visualizing the subject EvoStreets in one of three environments: 1) 2D orthographic projection with keyboard and mouse, 2) 2.5D projection with keyboard and mouse, and 3) virtual reality with head-mounted displays and hand-held controllers. The tasks were designed such that the participants had to visually examine and compare nodes and edges, where nodes represented files and edges related files sharing cloned code.

In order to identify differences between the three environments, we measured the time required to find an answer as well as its correctness for each task and environment. According to our results, we could not find statistically significant differences among the three different environments, though, there seems to be a trend that comparing blocks in 2.5D is more error prone than in 2D and VR. Among our 34 participants were 17 advanced undergraduate students who already have had experience with EvoStreets visualized in virtual reality. Comparing these students with the remaining participants, we neither could find differences regarding time and correctness. Based on our findings, we conclude that it cannot be assumed that 1) the 2D environment takes less time to find an answer, 2) the 2.5D and VR environments provide better results regarding the correctness of tasks compared to the 2D environment, and 3) the performance regarding time and correctness differs between the 2.5D and VR environments. Furthermore, each participant had the opportunity to rate each environment as to its usefulness to solve a certain task. On average, VR and 2.5D were rated similarly and better than 2D. Yet, the subjective assessment did not correlate with the actual task-performance measures.

To find differences in the navigation of keyboard and mouse versus controllers and head turning, we recorded the trajectories of each participant in the 2.5D and VR environments. We found that human beholders in VR tend to move to a lesser extent compared to the 2.5D environment. In future work, we want to continue analyzing the navigation paths of VR and 2.5D. That is, we want to study whether the differences between navigation result from using hand-held controllers or head-mounted displays or any other factor.

Acknowledgement: We want to thank all participants of our experiment and two of our research group members for their assistance in our study.

References

[1] F. Steinbrückner, "Consistent software cities: supporting comprehension of evolving software systems," Ph.D. dissertation, Brandenburgische Technische Universität Cottbus, 06 2013. [Online]. Available: https://opus4.kobv.de/opus4-btu/frontdoor/index/index/docId/1681

[2] R. Wettel and M. Lanza, "Visualizing software systems as cities," in *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*, June 2007, pp. 92–99.

[3] ——, "Codecity: 3d visualization of large-scale software," in *Companion of the 30th International Conference on Software Engineering*, ser. ICSE Companion '08. New York, NY, USA: ACM, 2008, pp. 921–922. [Online]. Available: http://doi.acm.org/10.1145/1370175.1370188

[4] ——, "Visual exploration of large-scale system evolution," in *2008 15th Working Conference on Reverse Engineering*, Oct 2008, pp. 219–228.

[5] D. Holten, "Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 741–748, Sep. 2006.

[6] B. Johnson and B. Shneiderman, "Tree-maps: A space-filling approach to the visualization of hierarchical information structures," in *Proceedings of the Conference on Visualization*. IEEE Computer Society Press, 1991, pp. 284–291. [Online]. Available: http://dl.acm.org/citation.cfm?id=949607.949654

[7] S. S. Chance, F. Gaunet, A. C. Beall, and J. M. Loomis, "Locomotion mode affects the updating of objects encountered during travel: The contribution of vestibular and proprioceptive inputs to path integration," *Presence: Teleoper. Virtual Environ.*, vol. 7, no. 2, pp. 168–178, 1998.

[8] J. Vincur, P. Navrat, and I. Polasek, "Vr city: Software analysis in virtual reality environment," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, July 2017, pp. 509–516.

[9] M. Rüdel, J. Ganser, and R. Koschke, "A controlled experiment on spatial orientation in vr-based software cities," in *IEEE Working Conference on Software Visualization*, Sep. 2018, pp. 21–31.

[10] M. Balzer, O. Deussen, and C. Lewerentz, "Voronoi treemaps for the visualization of software metrics," in *Proceedings of the 2005 ACM symposium on Software visualization*. ACM, 2005, pp. 165–172.

[11] C. Parnin, C. Görg, and O. Nnadi, "A catalogue of lightweight visualizations to support code smell inspection," in *ACM Symposium on Software Visualization*. New York, NY, USA: ACM, 2008, pp. 77–86. [Online]. Available: http://doi.acm.org/10.1145/1409720.1409733

[12] T. Bladh, D. A. Carr, and M. Kljun, "The effect of animated transitions on user navigation in 3d tree-maps," in *Ninth International Conference on Information Visualisation (IV'05)*, July 2005, pp. 297–305.

[13] R. Wettel and M. Lanza, "Visual exploration of large-scale system evolution," in *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*. IEEE, 2008, pp. 219–228.

[14] C. Knight and M. Munro, "Virtual but visible software," in *Information Visualization, 2000. Proceedings. IEEE International Conference on*. IEEE, 2000, pp. 198–205.

[15] N. Capece, U. Erra, S. Romano, and G. Scanniello, "Visualising a software system as a city through virtual reality," in *Augmented Reality, Virtual Reality, and Computer Graphics*, L. T. De Paolis, P. Bourdot, and A. Mongelli, Eds. Cham: Springer International Publishing, 2017, pp. 319–327.

[16] J. I. Maletic, J. Leigh, A. Marcus, and G. Dunlap, "Visualizing object-oriented software in virtual reality," in *International Workshop on Program Comprehension*, May 2001, pp. 26–35.

[17] T. Panas, R. Berrigan, and J. Grundy, "A 3d metaphor for software production visualization," in *International Conference Information Visualization*. IEEE, 2003, pp. 314–319.

[18] T. Panas, T. Epperly, D. Quinlan, A. Saebjornsen, and R. Vuduc, "Communicating software architecture using a unified single-view visualization," in *Engineering Complex Computer Systems, 2007. 12th IEEE International Conference on*. IEEE, 2007, pp. 217–228.

[19] F. Fittkau, A. Krause, and W. Hasselbring, "Exploring software cities in virtual reality," in *IEEE Working Conference on Software Visualization*. IEEE, 2015, pp. 130–134.

[20] P. Khaloo, M. Maghoumi, E. Taranta, D. Bettner, and J. Laviola, "Code park: A new 3d code visualization tool," in *IEEE Working Conference on Software Visualization*. IEEE, 2017, pp. 43–53.

[21] L. Merino, J. Fuchs, M. Blumenschein, C. Anslow, M. Ghafari, O. Nierstrasz, M. Behrisch, and D. A. Keim, "On the impact of the medium in the effectiveness of 3d software visualizations," in *IEEE Working Conference on Software Visualization*. IEEE, 2017, pp. 11–21.

[22] A. Schreiber and M. Brüggemann, "Interactive visualization of software components with virtual reality headsets," in *IEEE Working Conference on Software Visualization*. IEEE, 2017, pp. 119–123.

[23] K. Ogami, R. G. Kula, H. Hata, T. Ishio, and K. Matsumoto, "Using high-rising cities to visualize performance in real-time," in *IEEE Working Conference on Software Visualization*. IEEE, 2017, pp. 33–42.

[24] F. Fernandes, C. S. Rodrigues, and C. Werner, "Dynamic analysis of software systems through virtual reality," in *Symposium on Virtual and Augmented Reality*, Nov. 2017, pp. 331–340, in Spanish.

[25] A. Elliott, B. Peiris, and C. Parnin, "Virtual reality in software engineering: Affordances, applications, and challenges," in *IEEE/ACM International Conference on Software Engineering*, vol. 2, May 2015, pp. 547–550.

[26] D. Baum, J. Schilbach, P. Kovacs, U. Eisenecker, and R. Müller, "Getaviz: Generating structural, behavioral, and evolutionary views of software systems for empirical evaluation," in *IEEE Working Conference on Software Visualization*. IEEE, 2017, pp. 114–118.

[27] S. Diehl, Ed., *Software Visualization: International Seminar Dagstuhl Castle, Germany*, ser. Lecture Notes in Computer Science. Springer, 2001.

[28] S. Bassil and R. K. Keller, "A qualitative and quantitative evaluation of software visualization tools," in *Proceedings of the Workshop on Software Visualization*. IEEE, 2001, pp. 33–37.

[29] R. Koschke, "Software visualization in software maintenance, reverse engineering, and re-engineering: a research survey," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 15, no. 2, pp. 87–109, 2003.

[30] S. Carpendale and Y. Ghanam, "A survey paper on software architecture visualization," University of Calgary, Technical Report 2008-906-19, 2008.

[31] A. R. Teyseyre and M. R. Campo, "An overview of 3d software visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 1, pp. 87–105, 2009.

[32] S. Diehl, *Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. Springer, 2010.

[33] P. Caserta and O. Zendra, "Visualization of the static aspects of software: A survey," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 7, pp. 913–933, 2011.

[34] R. L. Novais, A. Torres, T. S. Mendes, M. Mendonça, and N. Zazworka, "Software evolution visualization: A systematic mapping study," *Information and Software Technology*, vol. 55, no. 11, pp. 1860–1883, Nov. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2013.05.008

[35] H. A. Basit, M. Hammad, and R. Koschke, "A survey on goal-oriented visualization of clone data," in *IEEE Working Conference on Software Visualization*. IEEE, 2015, pp. 46–55.

[36] D. A. Bowman, E. T. Davis, L. F. Hodges, and A. N. Badre, "Maintaining spatial orientation during travel in an immersive virtual environment," *Presence: Teleoper. Virtual Environ.*, vol. 8, no. 6, pp. 618–631, 1999. [Online]. Available: http://dx.doi.org/10.1162/105474699566521

[37] B. E. Riecke, D. W. Cunningham, and H. H. Bülthoff, "Spatial updating in virtual reality: the sufficiency of visual information," *Psychological Research*, vol. 71, no. 3, pp. 298–313, May 2007. [Online]. Available: https://doi.org/10.1007/s00426-006-0085-z

[38] J. W. Regian, W. L. Shebilske, and J. M. Monk, "Virtual reality: An instructional medium for visual-spatial tasks," *Journal of Communication*, vol. 42, no. 4, pp. 136–149, 1992. [Online]. Available: http://dx.doi.org/10.1111/j.1460-2466.1992.tb00815.x

[39] B. Sousa Santos, P. Dias, A. Pimentel, J.-W. Baggerman, C. Ferreira, S. Silva, and J. Madeira, "Head-mounted display versus desktop for 3d navigation in virtual reality: A user study," *Multimedia Tools and Applications*, vol. 41, no. 1, pp. 161–181, Jan. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11042-008-0223-2

[40] R. A. Ruddle, S. J. Payne, and D. M. Jones, "Navigating large-scale virtual environments: what differences occur between helmet-mounted and desk-top displays?" *Presence: Teleoperators & Virtual Environments*, vol. 8, no. 2, pp. 157–168, 1999.

[41] R. A. Ruddle and P. Péruch, "Effects of proprioceptive feedback and environmental characteristics on spatial learning in virtual environments," *International Journal of Human-Computer Studies*, vol. 60, no. 3, pp. 299 – 326, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1071581903001733

[42] P. Péruch, M. May, and F. Wartenberg, "Homing in virtual environments: Effects of field of view and path layout," *Perception*, vol. 26, no. 3, pp. 301–311, 1997.

[43] F. Wartenberg, M. May, and P. Péruch, "Spatial orientation in virtual environments: Background considerations and experiments," in *Spatial cognition*. Springer, 1998, pp. 469–489.

[44] R. F. Wang, "Between reality and imagination: When is spatial updating automatic?" *Perception & Psychophysics*, vol. 66, no. 1, pp. 68–76, Jan 2004. [Online]. Available: https://doi.org/10.3758/BF03194862

[45] R. Wettel, "Software systems as cities," Ph.D. Thesis, University of Lugano, Switzerland, 2010.

[46] M. R. Mine, "Virtual environment interaction techniques," University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, Tech. Rep., 1995.

[47] Y. Guan and M. Zheng, "Real-time 3d pointing gesture recognition for natural hci," in *2008 7th World Congress on Intelligent Control and Automation*, June 2008, pp. 2433–2436.

[48] M. Nielsen, M. Störring, T. B. Moeslund, and E. Granum, "A procedure for developing intuitive and ergonomic gesture interfaces for hci," in *International gesture workshop*. Springer, 2003, pp. 409–420.

[49] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady, "Evaluating display fidelity and interaction fidelity in a virtual reality game," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 4, pp. 626–633, 2012.

[50] L. E. Nacke, S. Stellmach, D. Sasse, and C. A. Lindley, "Gameplay experience in a gaze interaction game," *arXiv preprint arXiv:1004.0259*, 2010.

[51] C. K. Roy, M. F. Zibran, and R. Koschke, "The vision of software clone management: Past, present, and future (keynote paper)," in *Software Evolution Week—IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*, Feb 2014, pp. 18–33.

[52] H. A. Basit, M. Hammad, S. Jarzabek, and R. Koschke, "What do we need to know about clones? deriving information needs from user goals," in *International Workshop on Software Clones*. IEEE Computer Society Press, Mar. 2015, pp. 51–57.
[53] L. J. Cronbach and P. Meehl, "Construct validity in psychological tests," *Psychological Bulletin*, vol. 52, no. 4, pp. 281–302, 1955.