Collaborative software visualization with SEE

William Behnke
Dept. of Mathematics and Computer Science
University of Bremen
wbehnke@uni-bremen.de

Hannes Lennart Kuß
Dept. of Mathematics and Computer Science
University of Bremen
hkuss@uni-bremen.de

***Abstract*—SEE is a software engineering tool for visualizing software metrics based on the code-cities metaphor. It assists distributed teams in analyzing software collaboratively by offering multi-user functionality (including a voice chat), that allows team members to communicate naturally while investigating software. The tool utilizes graphs supplied in the Graph eXchange Language (GXL) format to represent software data, and allows users—among other things—to compare the current architecture with the original plan and to track changes of a software over time. One of our long-term goals is to enhance communication and collaboration among team members, to bridge spatial gaps, and to facilitate the understanding of software in (spatially separated) teams.**

I. Introduction

On a typical day, software developers spend around 82 minutes in meetings [1]. One of our goals in SEE is to reduce that time to the needed minimum. In software development, it is relevant to visualize different aspects of code in order to get a better understanding [2]. According to Cherubini et. al.: "developers produced visualizations: to understand, to design and to communicate." [2] However, the visualization of code is not easy and developers often miss this feature [2]. SEE (for Software Engineering Experience) is a tool developed by our research group to visualize software data in 3D using the code-city metaphor. [3].

Our tool focuses on co-operative understanding, enabling multiple people to understand and to communicate about software together. Our tool builds up on established 3D visualization platforms, specifically utilizing the Unity Engine. We empower developers to incorporate interactive capabilities, enabling multiple individuals to partake in virtual meetings, and facilitate information exchange within the context of software visualization.

SEE provides developers with enhanced abilities to comprehend the quality of their software, get a comprehensive overview, navigate complex architectural structures, and monitor the runtime behavior of their software. One of our goals is to support software development teams through the provision of an intuitive and informative environment. The development of SEE is being carried out at the University of Bremen in collaboration with Axivion [1], a company specialized in static code analysis and software architecture verification.

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

The focus of this paper is to introduce our project and, in particular, to emphasize the significance of incorporating emotional states within our software. Emotional state means the mood of the participating individual by analyzing their expressions and gestures.

In addition, we aim to present the future plans for our tool and provide further insight into the topic for other developers.

II. SEE Capabilities

The SEE platform is equipped with multiplayer functionality, which allows multiple users—represented as humanoid avatars—to concurrently interact with both the viewed software and each other. This includes actions such as moving the avatar or altering the location or size of components drawn within the *code city*, as well as the use of a voice chat to facilitate communication between users. This collaborative environment enables users to work together and discuss the software system being analyzed.

SEE represents a software system's architecture hierarchically (persisted as GXL graph), with each node in the graph representing a module of source code. Binary relations among components, such as function calls, are depicted as hierarchically bundled edges.

The incorporation of software metrics into the graph is achieved through the use of additional visual attributes of the shapes—such as depth, height, width or color—and additional decorations, e.g., antennas above blocks. The layout of the *Code city* is customizable and can be automated by different types of hierarchical graph layouts (e.g., treemaps, EvoStreets, rectangular and circular packing). An example of a *Code city* can be seen in Figure 1. The SEE platform enables—among others—the visualization of the current state of a software architecture through the representation of components as nodes and dependencies as edges connecting them. This representation can be compared to the original architecture plan, allowing users to identify discrepancies between the intended specification and the implemented architecture. This comparison facilitates a clear understanding of deviations from the original design and allows for the identification of potential issues or areas for improvement.

In addition to this, SEE offers an "evolution view" feature which allows users to view the historical evolution of a project, providing a comprehensive overview of the changes that occurred over time and identifying the

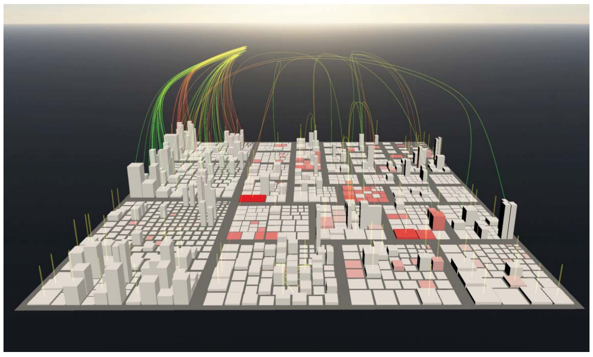

Fig. 1. Screenshot of an example *Code City*.

components that have undergone the most significant alterations. SEE also includes a built-in code viewer with the capability to synchronize the current line of code being viewed among all participants in a virtual meeting, enabling all attendees to easily reference the code being discussed.

III. Enhancing Nonverbal Communication

The SEE platform originated as a student project, initially utilizing the Unreal Engine as a base. After further development, we transitioned to the Unity Engine, as it offers greater versatility in terms of deployment across various platforms, and the ability to utilize a wider range of tools and community-driven products.

One of the most promising focus of development for the SEE platform is the improvement of non-verbal communication through capturing and displaying user emotions and gestures. To achieve this, we are experimenting with the use of the HTC Facial Tracker [2] and OpenCV [3] to monitor and mirror facial states. The incorporation of facial tracking technology into code review processes has several potential benefits. One of the key advantages is the ability to analyze nonverbal cues such as expressions and gestures, which can provide additional information about the emotional state of the individuals involved in the process.

In psychology it is well known, that nonverbal communication is as important as verbal communication. Hence, the incorporation of nonverbal communication in SEE can lead to more efficient and effective communication and collaboration among team members, as they are able to more easily interpret the reactions of their colleagues. Additionally, this may also aid in identifying areas of confusion or difficulty, allowing for more focused and productive discussions.

Furthermore, this feature can be especially beneficial for managers or supervisors who may not be profoundly involved in the technical aspects of the software development process. It allows them to more easily identify and address issues related to team dynamics and communication, improving the overall performance of the development team. By offering a clearer insight into the emotional state of the team, managers and supervisors can effectively address and resolve any conflicts or challenges that may arise during the development process, ultimately improving time management. In most project contexts, emotional states refer to the emotional well-being of software developers.

Overall, incorporating facial tracking technology into code review processes has potential to improve time management and increase the efficiency and effectiveness of software development teams. This can lead to more productive and successful software development projects, ultimately resulting in better software and more satisfied customers.

[2] see https://www.vive.com/de/accessory/facial-tracker/

[3] see https://opencv.org/opencv-face-recognition/

IV. Upcoming Features

In this paper, we discussed some key aspects for the continued development of SEE. Additionally, we also identified other areas of interest for further exploration, such as:

- *Improving edge visualization using different types of animations:* Improving the visualization of edges between different components of the software, by improving animations to more clearly represent relationships and dependencies.
- *Runtime configuration:* Providing more flexibility and control for users by allowing them to configure the visualization at runtime, tailoring it to their specific needs and preferences.
- *Live documentation:* Incorporating live documentation within SEE, to provide a more complete and integrated understanding of the software.

V. Conclusion

This paper presents SEE, a software visualization tool that facilitates software comprehension by means of Code Cities. We discussed the technology used in the development of SEE to demonstrate our plans for future implementations, and enhancing nonverbal communication by capturing and displaying user emotions and gestures. More information about SEE can be found on our website: https://see.uni-bremen.de/.

References

[1] A. N. Meyer, E. T. Barr, C. Bird, and T. Zimmermann, "Today was a good day: The daily life of software developers," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 863–880, 2021.

[2] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko, "Let's go to the whiteboard: How and why software developers use drawings," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 557–566. [Online]. Available: https://doi.org/10.1145/1240624.1240714

[3] R. Wettel and M. Lanza, "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: Association for Computing Machinery, 2008, p. 921–922. [Online]. Available: https://doi.org/10.1145/1370175.1370188