Is the 3D model the way to go when presenting microservice architecture?

Tomas Cerny
tcerny@arizona.edu
SIE, University of Arizona
Tucson, Arizona, USA

Amr S. Abdelfattah
amr_elsayed1@baylor.edu
SIE, University of Arizona
Tucson, Arizona, USA

Darek Gajewski
dgajewski@arizona.edu
SIE, University of Arizona
Tucson, Arizona, USA

Patrick Harris
patrick_harris3@baylor.edu
CS, Baylor University
Waco, Texas, USA

Mia Gortney
mia_gortney1@baylor.edu
CS, Baylor University
Waco, Texas, USA

Abstract

Software engineers are challenged with maintaining complex systems as the architecture becomes too difficult to understand in a 2D space. As a result, researchers look for system abstractions to help engineers understand the system architecture and its dependencies in both real time and statically. This problem is especially concerning to decentralized systems like those built with microservices. The advancement in the areas of virtual and augmented reality brings interesting directions for coping with complex models abstracting these systems. However, when considering a visual model to represent the system architecture with its dependencies, there are two most obvious options: an intractable two or three-dimensional model (2D/3D). This paper questions the effectiveness of virtual and augmented reality and 3D visual models to improve comprehension. In this paper, we develop two interactive visual models to represent microservice dependency graphs. One model is 2D, and the other is 3D, allowing us to render a microservice system with varying size. We then ask microservice developers questions about the abstracted system, and analyze the pros and cons of these models for the tasks related to system architecture comprehension.

CCS Concepts

• **Human-centered computing** → **Visualization**; • **Applied computing** → **Service-oriented architectures**; • **Software and its engineering** → **Software architectures**.

Keywords

Visualization, Program Comprehension, User Study, Microservices

ACM Reference Format:
Tomas Cerny, Amr S. Abdelfattah, Darek Gajewski, Patrick Harris, and Mia Gortney. 2024. Is the 3D model the way to go when presenting microservice architecture?. In *39th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW '24), October 27-November 1, 2024, Sacramento, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3691621.3694954

ASEW '24, October 27-November 1, 2024, Sacramento, CA, USA

ACM ISBN 979-8-4007-1249-4/24/10
https://doi.org/10.1145/3691621.3694954

1 Introduction

Maintainability of software systems has a significant impact on costs promoted to end users. A driving factor for system maintainability is robust software architecture. However, modern systems have become complex and decentralized, inviting multiple disjoint development teams to contribute to the overall software solution. A great example comes with microservices, which fuel cloud-native solutions. It is common for one microservice to be a manager but a single team [2] to support the separation of duty.

Increase in system complexity requires new methods to visualize the system as a whole. First, we condense two-dimensional (2D) representations; then, we add metadata to help illustrate more information. It has been established in other fields that 2D images suffer similar issues. Adding dimensions like time (real-time changes, growth) and space (the size of a component), along with other variables, further add complexities in a 2D visualization that can be difficult to differentiate. The paper starts from a proven theory that adding dimensions to the visual forms, allows humans to consume more data at once. Making it interactive allows engineers to ignore information to reduce noise.

To tackle the comprehension of dependencies in complex systems, we often use an abstraction model that represents selected architectural views [7]. A commonly used view for microservices is the service view, which shows the interconnection between services. Such a view is recovered through trace analysis or by a software architecture reconstruction process [3]. In the context of systems with a large number of microservices, proper visualization of the architectural view becomes critical as it impacts how quickly software engineers understand the system architecture and its properties.

In the context of visualization advancements, technologies that support virtual reality (VR) or augmented reality (AR) come into play. However, we must consider that these technologies influence the choice of visual models. To document software architecture, modeling languages like UML and SysML are commonly used, or ArchiMate is commonly used for enterprise architecture. However, all these models consider two-dimensional (2D) modeling aspects. For VR/AR, we would likely seek alternative visualization that would suit three-dimensional (3D) space.

The logical question to ask is how the 2D and 3D models compare when used by practitioners for tasks involving software architecture, its properties, and component dependencies. This paper

performs an experiment where a service view of a microservice system is rendered in two visual models, and microservice engineers participate in a user study that asks them to extract common system architectural properties from the two different visualizations. It analyzes the models as instruments to aid engineers in extracting correct answers about the system based on a set of questions; it also considers their satisfaction and feedback. The controversial findings are elaborated on and discussed in this paper.

This paper is organized as follows. Section 2 introduces background. Section 3 details the experiment and user study. Section 4 provides a discussion, and Section 5 concludes the paper.

2 Background

An architecture description is formally described by the ISO/IEC/IEEE 42010:2022 [7]. It describes the structure and expression of various entities, including software, systems, enterprises, systems of systems, etc. In this description, we consider that architecture serves many purposes, and for this reason, we can consider multiple viewpoints. An architecture viewpoint is a way of looking at a system of interest. To illustrate this, if we were in a custom house construction business, there would be different interests for an electrician and a plumber. When we consider a specific system, the particular viewpoint leads to an architectural view that might address multiple concerns. For instance, for a particular house, we might produce a sketch for a plumber; however, even an electrician might use that sketch to ensure proper distances for his work. The architectural view thus exposes architectural descriptions of a system of interest. Using this form of abstraction has the benefit of focusing on specific concerns or stakeholder interests. An architecture description is a work product that expresses an architecture. It is used to help stakeholders understand, analyze, and compare architectures. An architecture description may take the form of document(s), model(s), simulation(s), or other forms.

In the context of this paper, a **visual model** is our interest. When we consider software systems, the traditional way to design systems has been using visual models [14], such as UML, SysML, or ArchiMate. The advantage of ArchiMate is that it is meant for enterprise systems, while UML suite models small system parts. Both of these models operate in the 2D space [3]. ArchiMate is typically modeled at four levels with different specializations: Business, Application, Data, and Technology, which to some extent correspond to the architectural views that have also been used for microservices [9].

As mentioned in the introduction, the specific perspective considered in this paper relates to microservices. The service viewpoint is considered the most common visualization for microservices in literature [5]. This viewpoint represents interconnection across services that typically interact through remote calls to exposed endpoints. However, it is also possible to use message queues with implicit invocation as well. Established tools typically use a 2D graph where nodes represent a microservice and oriented edges show the direction of the interconnection implied by remote calls. Such graphs are reconstructed from dynamic traces by monitoring or through software architecture reconstruction [3].

A few studies have compared different visualizations for microservice architecture viewpoints, i.e., the AR vs. 2D in [1]. However, these studies did not provide a fair comparison between abstract visual models. Notably, there is a lack of research in the community to determine if the 3D model's superiority can be naturally assumed simply because it utilizes an additional dimension.

This paper develops two visual models with equal levels of interactivity and rendering space to assess the understandability of service viewpoint across these dimensions. This approach ensures a more accurate assessment of both 2D and 3D models, addressing whether the 2D or 3D model is the preferred choice for visualizing microservices.

3 Experiment

The objective of this experiment is to assess how the choice between 2D and 3D models impacts practitioners' efficiency in understanding the architectural properties of microservice systems.

We measure the impact of the visual dimension of the visualization on the system's understandability and examine the suitability of each model for rendering different system variants. In summary, the goal of the experiment is formulated as follows:

> Evaluating two visualization approaches, for the purpose of measuring the microservice system *understandability*, under the control of the same level of interactivity.

We ask the following research questions (RQs) to achieve the goal:

- RQ1: Is 2D or 3D visualization more efficient for understanding microservice systems?
- RQ2: How do participants perceive the use of 2D/3D models for assigned tasks?
- RQ3: What are the challenges identified by participants regarding the 2D/3D visualization techniques?

3.1 Benchmark and Visualization Design

To set realistic settings, we use a well-established system benchmark called TrainTicket [13] that has been cited over 150 times on Google Scholar. This benchmark represents a train ticket registration system and consists of 47 microservices. This system uses remote procedure calls, and we reconstructed the service dependency graphs (SDG) using static analysis of the source code, similarly as described in [3], to avoid manual errors. The intermediate representation of the SDG has been extracted from 44 microservices (as we were constrained by Java) in the form of a JSON file. Additionally, we anonymized and augmented the microservice names to resemble different systems to minimize the learning effect in the study.

We implemented two proofs-of-concept (PoC) tools [6] to visualize microservice systems from the JSON intermediate representation of the SDG. Figs. 1 and 2 show the visual models provided by our PoCs that implement the 2D and 3D versions of the SDGs. In the SDG, microservices are presented as nodes, and the dependency relationships between them are represented by links connecting the nodes. Dependency relationships were extracted using static analysis [4, 11] and, in particular, analyzing the REST calls sent between services. The size of the links denotes how many endpoints

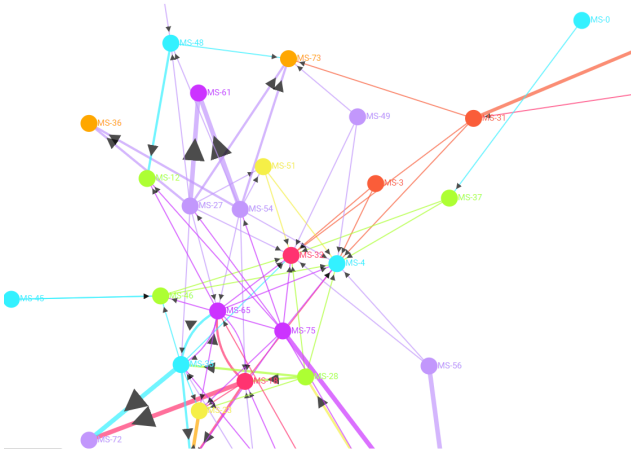

Figure 1: A 2D representation of multiple nodes connected.

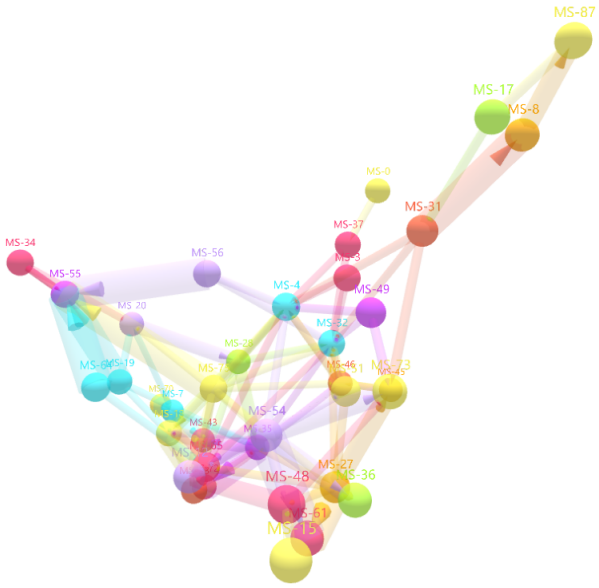

Figure 2: A 3D representation of multiple nodes connected.

comprise that specific link. The more endpoints connecting the two nodes, the thicker the link.

The goal was to reduce the variability between the tools so that when comparing them, the abundance or lack of features was not a factor in our analysis. Our PoC implemented several features to increase the system's usability. The nodes in the system can be dragged into any position the user desires. The view of the graph itself can also be explored through panning, zooming, and rotating (in the case of the 3D graph). When the user clicks on a node, an info box pops up showing the node/microservice name and type, as well as its dependencies and dependents of other nodes/microservices. Each shows the contributing REST calls listed as attributes. Upon a right-click on a node, there is an option to isolate the node to only its neighbors.We also implemented an interactive search, which was pointed out as a major challenge in an AR study mentioned in related work [1]. Each node's color is distinct and randomly assigned, with the node and its outgoing links sharing the same color.

3.2 Participants

We reached out to ten engineers from Europe and the U.S. as potential participants. They were asked to complete a pre-study survey to assess their expertise with microservices. Based on the survey, 6 engineers were selected for the experiment. These participants were all industry professionals with an average of 5 years of experience working with microservices in various job roles.

3.3 Artifacts

The experiment tasks were designed and delivered via Google Forms, with PoC visualization tools deployed and web links provided to participants for both the 2D and 3D visualizations. The tasks included nine questions about the system, requiring engagement with the visualization PoC. These questions were categorized into specific areas: identifying incoming and outgoing dependencies (2 questions), coupling and endpoint identification (3 questions), endpoint analysis (2 questions), and identifying cycles (2 questions).

The training material consisted of PowerPoint slides that covered the PoC's features and interaction methods. Each slide included a link to a deployed version of the PoC with a simple dataset, encouraging hands-on practice.

A feedback form, created using Google Forms, was used to gather participant impressions. This form featured two types of questions: participants rated various aspects of the visualization on a 5-point Likert scale, quantifying their experience and evaluating ease of use, completeness, and efficiency. Additionally, participants provided open-ended feedback on what they liked most and least about each PoC and suggested improvements. These artifacts collectively guided participants through tasks, training, and feedback processes, ensuring comprehensive data collection and analysis.

3.4 Design Procedure

The study was designed as a between-subjects experiment. Each participant received a script through e-mail divided into two sections: one for the 2D visualization and one for the 3D. Participants were randomly assigned, so half began with the 3D model and the other half with the 2D to minimize learning effects. The study was structured to be completed within 60 minutes, with 10 minutes allocated for training, 15 minutes for tasks, and 10 minutes for feedback for each visualization. Participants were instructed to submit their completed task form if they reached the 15-minute mark.

Participants began by completing the training material. They then recorded their screens while performing the tasks, providing insight into their interaction with the PoC. Following this, participants were required to use the PoC to complete the task form. They accessed a deployed version of the visualization PoC with an obfuscated TrainTicket benchmark via a provided link. After each task

group, participants reported the time taken to complete the group, aiding in the assessment of PoC difficulty. Finally, participants filled out a feedback form accessed via a provided link.

3.5 Analysis Approach

The participant task answers and feedback were analyzed towards accuracy and perception of the system service view in 2D/3D models.
RQ1: Participants were asked to complete nine categorized tasks. We prepared predetermined correct answers for each task, allowing us to compare participant responses and calculate a correctness ratio (0-1) for each task. Additionally, at the end of each task category, participants reported the recorded the time on their stopwatch. This allowed us to measure both the accuracy of their answers and the time taken to complete each set of tasks.
RQ2: The perception of the systems was evaluated using a feedback form completed by participants after finishing the tasks. The form included a series of questions measured on a 5-point Likert scale, assessing aspects like usability, understandability, completeness, and accessibility. Participants also rated the ease of navigating the graph and the usefulness of features like zooming, panning, and rotating. We calculated descriptive statistics for these responses to describe participants' perceptions of the two different visualizations.
RQ3: The verbalized challenges in both visualizations were evaluated through feedback analysis. In the feedback form, participants were asked which features were most useful and what they would improve or change about the visualization. They also described any difficulties encountered. These responses besides the analysis of recordings helped identify the challenges in both approaches. We used the Open and Axial qualification methodologies [8, 10] to analyze and categorize the challenges faced by the participants.

3.6 Data Analysis Results

We analyzed the collected data from participants and analyzed them for the corresponding data to answer the RQs.
RQ1: According to the study results, there does not appear to be a significant advantage between 2D and 3D visualizations. Participants were asked to answer nine questions for both the 2D and 3D systems. The overall answer correctness for the 2D visualization was 67%, while the 3D had a very similar accuracy of 65%. The most accurate answers were for the Incoming and Outgoing Dependents categories in both visualizations, with nearly 100% correctness. However, the least accuracy was observed in the Endpoint Analysis and Cyclic Dependencies categories, with around 40% correctness.

When analyzing the time taken to answer, the 3D visualization required more time for participants to analyze the system and answer most of the questions compared to the 2D model. The average total time for the 2D visualization was 11:12 minutes, whereas the 3D visualization took 14:51 minutes. The most time-consuming categories were Coupling and Endpoint Identification, with participants taking an average of 3:54 minutes in 2D and 6:33 minutes in 3D. Similarly, for the Endpoint Analysis category, the 2D visualization took 2:20 minutes, while the 3D visualization took 3:45 minutes. The other categories showed marginal timing differences.
RQ2: Overall, participants seemed to find the 2D PoC more favorable than the 3D. Participants were almost twice as likely to recommend the 2D PoC for daily work over the 3D. 50% of the participants found navigating the 2D model to be easy or very easy compared to only 16.7% for the 3D model. Only 16.7% of the participants found manipulating the placement of nodes in the 2D model to be not useful or not useful at all compared with 33% on the 3D model. Participants consistently found the 2D PoC to have more usability, more understandability, more quick to get needed information, and more complete.
RQ3: The challenges were categorized into Visual Decisions, Navigation Decisions, and Interactivity. Regarding navigation decisions, one challenge participants voiced is related to both visualizations. Participants often spatially oriented themselves in a system based on the current context of what they were viewing (coordinates, level of zoom, nodes in view, etc.). When interactions changed the context of the system they were viewing, some participants found themselves disoriented and confused about where they were in the system versus where they were. This is most evident when clicking on a node, which will bring up an informational box about that node but will also zoom in on that node. Closing the informational box then zooms back out to the overall system. These decisions made it hard for participants to maintain a certain context within the system and led to overall confusion.

The visual presentation of the PoC is important for conveying information about the system in a useful and easy-to-understand manner. Thus, when the visual presentation is lacking in some areas, it may be harder to use the PoC to achieve the purpose of analyzing and understanding a microservice-based system. Participants pointed out areas where the decisions we made led to a poor visual presentation. In 2D, one complaint was some of the lighter colors of nodes, and node names made them harder to pick out against the white background. However, in 3D, the complaint was that the arrows were too small, making it hard to determine the correct direction of a dependency relationship.

A final challenge concerned interactivity. Our PoC aimed to be as versatile, fluid, and interactive as possible by providing many features to better analyze the system. However, some found the amount of interactivity overwhelming, especially in 2D models.

3.7 Threats to Validity

In terms of validity threats [12], poor training materials could confuse participants about the PoC and its use. Therefore, we had external reviewers assess the materials and conduct a trial run before distribution. Participants also rated the training material in their feedback forms. Additionally, unclear task questions might lead to incorrect results; therefore, we used appropriately detailed language for our target audience and tested the study on colleagues unfamiliar with the PoC. Using a single system benchmark might misrepresent dependencies, so we employed an established benchmark from the scientific community. While there was a risk of incorrect SDG construction, we manually verified its accuracy. All visualized system variants were obfuscated forms of the benchmark.

The small participant pool may limit generalizability, as individual responses have a larger impact on the results. To address this, we used a between-subject design, effectively doubling the participant count. Also, we employed different anonymized names for microservices in both visualizations, ensuring that participants were not exposed to the same version, which helped reduce bias.

4 Discussion

The results of this study indicate that 3D visual models do not outperform 2D visual representations in the area of microservice architecture and program comprehension. The practitioners seem to slightly prefer 2D visual models. One must consider the practicality of 3D models where practitioners place themselves in front of Integrated Development Environments and have mostly been exposed to static models or those rendered in 2D. The 3D model perspective, which is likely preferred for VR/AR, will need to overcome the burden that all practitioners have been educated in 2D models, and the 3D perspective becomes a new instrument to them as they have not used it before. There must be a convincing argument for the 3D perspective to engage engineers in finding benefits in it over efficient and interactive 2D models. When discussing our results with non-engineers, a similar point has been made that individuals use 2D models in their education, and this can greatly influence their choice when offered an alternative. We must admit that we saw it as a disappointment at first that the 3D model was seen as less suitable for the task, but likely the trajectory to us it might lead through 3D education, and thus, younger generations of engineers might be more engaged with it.

Interestingly, the related study by Abdelfattah et al. [1] revealed that AR visualization is better suited to the microservice architecture viewpoint compared to static 2D visualization. Although there are multiple differences between that study and ours, it raises an important point: AR could potentially promote the 3D model to outperform 2D visualization, even if it is interactive. This opens a new challenge, suggesting that next we should assess the impact of AR/VR mediums on the 3D model and conduct a study to compare it with interactive 2D visualization.

5 Conclusion

3D visualization might not have the expected advantages when compared to the 2D approach while targeting engineers interested in understanding the architecture of microservice systems. Although more time may be needed to adjust to navigating in 3D space on a 2D screen, there is little impact on system understandability, whether a visualization uses 2D or 3D models. Our experiment showed that no matter the model, the answers maintained similar correctness from participants. However, for 3D, correctness comes at the cost of a greater time taken to analyze the system over the 2D model. Practitioners in our experiment preferred 2D models, likely due to their educational background and greater familiarity with this format. In future work, we aim to study the impact of the AR/VR medium on the efficiency of these visualizations.

Data Availability

The PoC source codes, tools' screenshots, protocol material, and study results are publicly available at https://zenodo.org/records/12798890.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. OISE-1854049 and OISE-2409933.

References

[1] Amr S Abdelfattah, Tomas Cerny, Davide Taibi, and Sira Vegas. 2023. Comparing 2D and Augmented Reality Visualizations for Microservice System Understandability: A Controlled Experiment. In *2023 IEEE/ACM 31st International Conference on Program Comprehension (ICPC)*. 135–145.

[2] Dario Amoroso d'Aragona, Xiaozhou Li, Tomas Cerny, Andrea Janes, Valentina Lenarduzzi, and Davide Taibi. 2023. One microservice per developer: is this the trend in OSS?. In *European Conference on Service-Oriented and Cloud Computing*. Springer Nature Switzerland Cham, 19–34.

[3] Tomas Cerny, Amr S Abdelfattah, Jorge Yero, and Davide Taibi. 2024. From static code analysis to visual models of microservice architecture. *Cluster Computing* (2024), 1–26.

[4] Tomas Cerny, Amr S. Abdelfattah, Jorge Yero, and Davide Taibi. 2024. From static code analysis to visual models of microservice architecture. *Cluster Computing* 27, 4 (2024), 4145–4170. https://doi.org/10.1007/s10586-024-04394-7

[5] Mia E Gortney, Patrick E Harris, Tomas Cerny, Abdullah Al Maruf, Miroslav Bures, Davide Taibi, and Pavel Tisnovsky. 2022. Visualizing microservice architecture in the dynamic perspective: A systematic mapping study. *IEEE Access* (2022).

[6] Patrick Harris, Mia Gortney, Amr S. Abdelfattah, Tomas Cerny, and Pablo Rivas. 2024. Designing a System-Centered View to Microservices Using Service Dependency Graphs: Elaborating on 2D and 3D visualization. In *2024 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)* (Maldives).

[7] ISO/IEC/IEEE 42010 2022. *Software, systems and enterprise — Architecture description*. Standard. International Organization for Standardization, Geneva, CH.

[8] Shahedul Huq Khandkar. 2009. Open coding. *University of Calgary* 23, 2009 (2009), 2009.

[9] Florian Rademacher, Sabine Sachweh, and Albert Zündorf. 2020. A Modeling Method for Systematic Architecture Reconstruction of Microservice-Based Software Systems. In *Enterprise, Business-Process and Information Systems Modeling*. Springer International Publishing, Cham, 311–326.

[10] Maike Vollstedt and Sebastian Rezat. 2019. An introduction to grounded theory with a special focus on axial coding and the coding paradigm. *Compendium for early career researchers in mathematics education* 13, 1 (2019), 81–100.

[11] Andrew Walker, Ian Laird, and Tomas Cerny. 2021. On Automatic Software Architecture Reconstruction of Microservice Applications. In *Information Science and Applications*, Hyuncheol Kim, Kuinam J. Kim, and Suhyun Park (Eds.). Springer Singapore, Singapore, 223–234.

[12] C. Wohlin, P. Runeson, M. Host, M.C. Ohlsson, B. Regnell, and A. Wesslen. 2000. *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Norwell, MA, USA.

[13] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chenjie Xu, Chao Ji, and Wenyun Zhao. 2018. Benchmarking microservice systems for software engineering research. In *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*. 323–324.

[14] Zhengshu Zhou, Qiang Zhi, Shuji Morisaki, and Shuichiro Yamamoto. 2020. A Systematic Literature Review on Enterprise Architecture Visualization Methodologies. *IEEE Access* 8 (2020), 96404–96427. https://doi.org/10.1109/ACCESS.2020.2995850