Visualizing and Exploring OSGi-based Software Architectures in Augmented Reality

Extended Abstract

Artur Baranowski
German Aerospace Center (DLR)
Köln, Germany
artur.baranowski@dlr.de

Peter Seipel
German Aerospace Center (DLR)
Köln, Germany
peter.seipel@dlr.de

Andreas Schreiber
German Aerospace Center (DLR)
Köln, Germany
andreas.schreiber@dlr.de

ABSTRACT

This demo presents an immersive augmented reality solution for visualizing OSGi-based software architectures. By employing an island metaphor, we map abstract software entities to tangible real-world objects. Using advanced input modalities, such as voice and gesture control, our approach allows for interactive exploration and examination of complex software systems.

CCS CONCEPTS

• **Human-centered computing** → **Information visualization**; **Mixed / augmented reality**; *Interface design prototyping*;

KEYWORDS

Software visualization, real-world metaphor, augmented reality, user interface design

ACM Reference Format:
Artur Baranowski, Peter Seipel, and Andreas Schreiber. 2018. Visualizing and Exploring OSGi-based Software Architectures in Augmented Reality: Extended Abstract. In *VRST 2018: 24th ACM Symposium on Virtual Reality Software and Technology (VRST '18), November 28-December 1, 2018, Tokyo, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3281505.3281564

1 INTRODUCTION

As software systems expand in functionality, their complexity rises, making the underlying architecture increasingly obscure. Interactive visualizations play an important role in making large software projects more accessible [6]. By employing real-world metaphors, abstract software entities are mapped to tangible objects, such as buildings or islands. Common examples are cities [7] or cartographic metaphors [2]. These approaches to software architecture visualization allow engineers to quickly comprehend the purpose of software components and the dependencies between them.

Improvements in immersive Virtual Reality (VR) technology enable new ways of visualizing software architectures [3]. In context of software architecture visualization however, Augmented Reality (AR) solutions have crucial advantages over comparable VR applications. They usually provide see-through visors, which allow for an unobstructed view of the surrounding real-world environments. This way, classical desktop interfaces with text-based code-editing tools can be used in combination with immersive AR devices. Moreover, peers can communicate seamlessly when exploring software visualizations in AR collaboratively, because verbal and non-verbal means of communication remain intact.

We demonstrate an interactive software visualization application using the immersive AR head-mounted display Microsoft HoloLens. We visualize OSGi-based software projects using an island metaphor [3].

2 VISUAL METAPHOR

Our approach employs an *island metaphor* [3] for visualizing OSGi-based software architectures. Within the confines of a virtual tabletop, a software system is represented as an archipelago. Software artifacts can be explored at different levels of granularity (Figure 1). At the highest level, bundles are mapped to individual islands. Islands are divided into multiple regions, representing the packages within a bundle. Finally, each compilation unit contained in a package is represented as building within a corresponding region. The size of an island is proportional to the number of a bundle's compilation units, ensuring all buildings can be accommodated without overlapping.

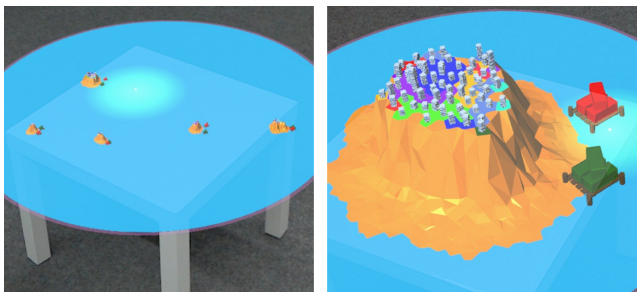

Figure 1: Visualizing software artifacts at different levels of granularity.

3 TECHNICAL DESCRIPTION

The Microsoft HoloLens is an optical see-through head-mounted mobile system. As such, it features an array of sensors, enabling it to capture gesture and voice input. Moreover, it is capable of tracking its position and orientation in 3D space. The standard runtime environment for HoloLens applications is the Universal Windows

VRST '18, November 28-December 1, 2018, Tokyo, Japan

ACM ISBN 978-1-4503-6086-9/18/11.
https://doi.org/10.1145/3281505.3281564

Platform (UWP). The Unity3D game engine provides specific tools for UWP development. Additionally, the Mixed Reality Toolkit[1] (MRTK) for Unity offers a collection of scripts to facilitate HoloLens application development in Unity3D.

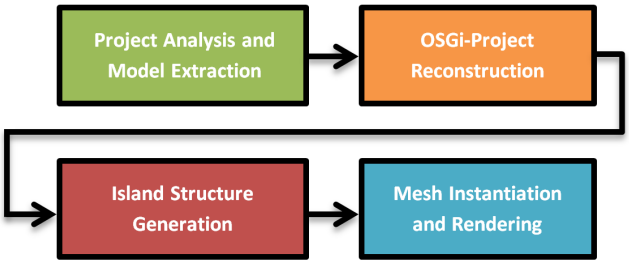

Figure 2: Visualization construction process.

First, we extract all relevant pieces of information regarding a particular software project into a JSON file [4]. On the HoloLens, this file is read into an in-memory data structure. It can be queried hierarchically and is used to retrieve and reconstruct the OSGi project. Subsequently, based on the number of packages and compilation units contained within each bundle, island structures and their positions are calculated. Each island is assigned an individual Voronoi diagram. Regions are formed by iteratively claiming cells from that diagram. Finally, based on the calculated island structures, we generate the island polygon meshes and all objects contained, such as buildings and regions.

4 INTERACTION

We use a state machine, combining HoloLenses three main input modalities *Gesture*, *Voice*, and *Gaze*, to provide a context-sensitive interface governing the possible interactions depending on the applications current state.

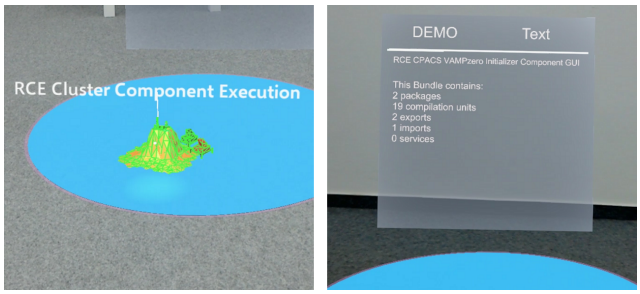

Figure 3: For a selected bundle specific information is displayed in a information panel.

4.1 Gesture interaction

The gesture control component enables both one-handed and two-handed input. By performing an "Air-Tap"-Gesture[2], a bundle can be selected. A selected bundle is enclosed by a green wire-frame, giving the user visual feedback about the selection (Figure 3). An information panel appears, giving the user bundle-specific information. By focusing the gaze-cursor on the virtual tabletop and performing a "Tap-and-Hold"-Gesture, the archipelago can be navigated. By performing a two-handed "Tap-and-Hold"-Gesture and dragging the hands apart, the visualization can be zoomed.

4.2 Voice Control

To enable voice control without using commands based on keywords, we implemented a natural language understanding component consisting of two parts [5]. The first parts task is to convert a natural language string into a user intention. To exonerate the HoloLens from additional processing overhead, this service is implemented as a web server application using the Rasa chatbot library [1]. Rasa uses machine learning to increase the number of statements connected to an intention. The second part contains the speech-to-text process. It sends the converted string as a request to the server. Furthermore, it processes the server's response.

5 COLLABORATION

Using the Mixed Reality Toolkit's sharing API, application states can be shared over multiple devices. Like this, we enable collaborative exploration of software architectures. Based on the visualizations, co-located peers can discuss design decisions or introduce new colleagues to the project. For the purpose of this demo, we employ a simple collaborative technique, where only one user at a time can perform interactions. As long as a user performs an interaction, all other users are temporarily blocked from interacting with the system.

6 CONCLUSION

We proposed an immersive and interactive AR software visualization solution for OSGi-based projects. Future work will include visualizations of bundle dependencies, as well as referencing and providing relationships between service components and service interfaces. Moreover, we will improve on the collaborative metaphor.

REFERENCES

[1] Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol. 2017. Rasa: Open Source Language Understanding and Dialogue Management. *CoRR* abs/1712.05181 (2017). arXiv:1712.05181

[2] Adrian Kuhn, Peter Loretan, and Oscar Nierstrasz. 2008. Consistent Layout for Thematic Software Maps. In *2008 15th Working Conference on Reverse Engineering*. 209–218.

[3] Andreas Schreiber and Martin Misiak. 2018. Visualizing Software Architectures in Virtual Reality with an Island Metaphor. In *Virtual, Augmented and Mixed Reality: Interaction, Navigation, Visualization, Embodiment, and Simulation*, Jessie Y.C. Chen and Gino Fragomeni (Eds.). Springer International Publishing, Cham, 168–182.

[4] Doreen Seider, Andreas Schreiber, Tobias Marquardt, and Marlene Brüggemann. 2016. Visualizing Modules and Dependencies of OSGi-Based Applications. In *2016 IEEE Working Conference on Software Visualization (VISSOFT)*. 96–100.

[5] Gokhan Tur and Li Deng. 2011. *Intent Determination and Spoken Utterance Classification.* Wiley-Blackwell, Chapter 4, 93–118.

[6] Colin Ware and Glenn Franck. 1996. Evaluating Stereo and Motion Cues for Visualizing Information Nets in Three Dimensions. *ACM Trans. Graph.* 15, 2 (April 1996), 121–140.

[7] Richard Wettel and Michele Lanza. 2007. Visualizing Software Systems as Cities. In *2007 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis*. 92–99.

[1] https://github.com/Microsoft/MixedRealityToolkit-Unity
[2] https://docs.microsoft.com/en-us/windows/mixed-reality/gestures