Immersive Exploration of OSGi-based Software Systems in Virtual Reality

Martin Misiak* TH Köln
Doreen Seider† German Aerospace Center (DLR)
Sascha Zur† German Aerospace Center (DLR)
Arnulph Fuhrmann* TH Köln
Andreas Schreiber† German Aerospace Center (DLR)

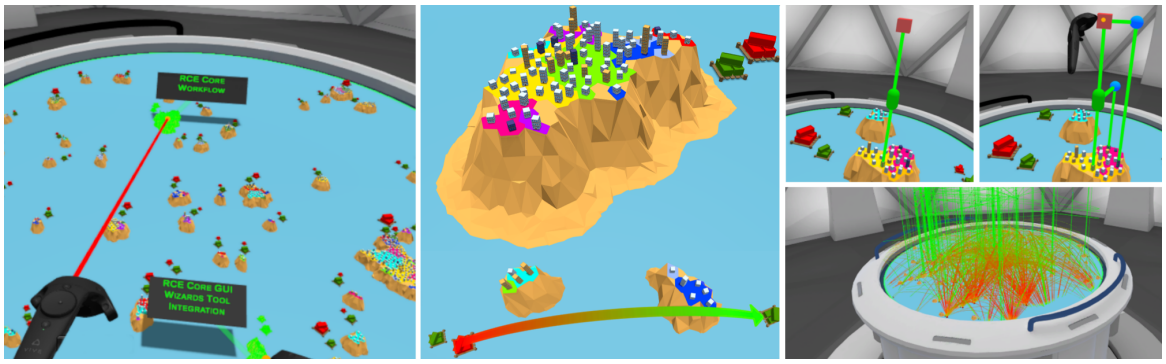

Figure 1: The software visualization application *IslandViz* for OSGi-based systems in virtual reality. (left) Software modules are represented as islands. (middle) Islands have very distinct shapes. Dependencies between them are visualized with arced arrows. (right top) Service connections are routed via nodes and distributed over the available height. (right bottom) All service and bundle dependencies of a complex software system are shown simultaneously inside the bounds of a virtual table.

Abstract

We present an approach for exploring OSGi-based software systems in virtual reality. We employ an island metaphor, which represents every module as a distinct island. The resulting island system is displayed in the confines of a virtual table, where users can explore the software visualization on multiple levels of granularity by performing intuitive navigational tasks. Our approach allows users to get a first overview about the complexity of an OSGi-based software system by interactively exploring its modules as well as the dependencies between them.

Keywords: Software visualization, OSGi, real-world metaphor, virtual reality.

1 Introduction

With increasing functionality, the complexity of a software system grows and hinders its further development. Visualization techniques can reduce the perceived complexity and aid the comprehension process. Over the years a number of two and three dimensional visualization approaches have been proposed. However, the visualization of software in *virtual reality (VR)* still remains a sparsely researched field. Its main advantages are the reduction of navigational difficulties encountered in classical 3D visualizations, stereoscopic depth cues and more intuitive interaction possibilities.

We present an approach for visualizing OSGi-based software systems in VR using an island metaphor. Our goal is to aid the high level comprehension of such systems, while minimizing the experienced simulator sickness and enabling an intuitive navigation.

OSGi (Open Services Gateway Initiative), is a module-based, service-oriented framework specification for Java. It centers on application development using modular units, called *bundles*. A bundle is a self-contained unit of classes and packages, which can be selectively made available to other bundles via import/export declarations. Popular implementations are *Apache Felix* or *Equinox*.

*e-mail: first name.last name@th-koeln.de
†e-mail: first name.last name@dlr.de

2 Related Work

In recent years, the software visualization field has made extensive use of real-world metaphors. Among the most frequently used is the city metaphor, which was influenced by the work of Wettel et al. [10]. Although we use a different metaphor, we share its building-based class representation. Kuhn et al. [2] cluster software artifacts, based on lexical similarity, to create 2D cartographic software maps. While the cartographic metaphor is similar to ours, we create distinct islands, based on hierarchical information.

Fittkau et al. [1] proposed an approach for a live trace visualization using a city metaphor. The visualization is displayed inside the *Oculus Rift DK1*, while the interaction is gesture based and uses a gaze driven pointer. Schreiber et al. [5] proposed an approach for visualizing software modules in VR, using an electrical component metaphor. Modules are represented as blocks and the containing packages are stacked on top of each module. Merino et al. [3] and Vincur et al. [8] presented a VR visualization for object oriented software using a city metaphor. The approaches rely on physical movement as their main navigational mechanism. In contrast, our work uses an explicit transformation of the visualization itself and is therefore independent of the available tracking space.

3 Islands

The visualization metaphor has to be expressive enough to provide mappings for a number of software artifacts. As pointed out by Wettel et al. [10], classes are the cornerstones of the object-oriented paradigm and are therefore the finest granular artifacts we aim to visualize. In the case of OSGi-based software, the metaphor needs to account for the following artifacts: class types, packages, bundles, service components, and service interfaces. We are interested in the import and export relations of individual bundles, as well as the referencing and providing relationships between service components and their respective interfaces. The metaphor should also

emphasize the bundle layer, as it forms a central aspect of OSGi.

We propose an island metaphor for the visualization of OSGi-based software systems. The entire software system is represented as an ocean with many islands on it. Each island represents an OSGi bundle and is split into multiple regions. Each region represents a Java package and contains multiple buildings. The buildings are the representatives of the individual class types which reside inside of a package. Each region provides enough space to accommodate all of its buildings without overlapping, and hence the overall size of an island is proportional to the number of class types inside of a bundle. In order to emphasize the plausibility of the metaphor, our islands strive for a high resemblance with real-world islands.

The island construction is based on claiming cells in a Voronoi diagram and is analogous to the work of Yang et al. [11]. Due to the probabilistic nature of this algorithm, the islands acquire a very distinct and rugged shape, which allows for a better memorability and thus, reduces the dependency on name labels. To maximize the visibility of class types, a multi-storey building representation is chosen, encouraging a metric based expansion in the height dimension. Our prototype uses the Lines of Code metric, where for every n lines of code, a storey is added to the building.

3.1 Visualization of Dependencies

Due to motion parallax and stereoscopic depth cues, VR strengthens the comprehension of relational information [9]. This makes it particularly well suited for visualizing dependencies.

3.1.1 Bundle Dependencies

Building on the island metaphor we use ports, situated along the coast line, which manage the incoming and outgoing dependencies. In order to avoid island intersection, the dependencies are visualized as vertically arced arrows. A color gradient, together with the arrow head indicate the dependency direction. The width is mapped to the number of packages which are being imported or exported over the given connection.

3.1.2 Service Dependencies

The main entities of the OSGi service layer are service interfaces and service components. As these components are linked to Java class types, we visualize them as special building types. A service component can reference as well as provide for multiple service interfaces. We represent these relationships with a straight line connection. However to avoid intersections and to leverage the vacant height dimension, we reroute the connections through special nodes, which are distributed evenly across the available height. They are located directly above the buildings in question and indicate what type of relationship is present.

4 Integration into a VR Environment

The entire software visualization is displayed in the confines of a virtual table. Although this imposes a restriction on the size of the usable visualization space, it offers several advantages. When inspecting the software, the user does not experience any relocation, since only the visualization in the confinements of the table has to be changed without altering the virtual room around it. This reduces user disorientation and simulator sickness, as the room always provides a stable frame of reference [4]. Additionally, the user does not have to move around excessively in the virtual environment to view a desired information. This allows for a standing as well as a seated VR experience.

In order to inspect arbitrary large software inside of the table, the user can manipulate the visualization itself. Our manipulation technique encompasses translation, rotation and scaling and is very similar to the Two-Handed Interface technique described by Schultheis et al. [6]. In contrast to their work we constrain the rotation to the up axis. The scaling operation is especially important, as zooming is directly tied to the transition between the individual abstraction layers (island, region, building) of the software system. This mode of interaction basically follows a level of detail scheme, where the elements belonging to a specific layer can be interacted with, as soon as they are large enough for the user to see and select.

5 Implementation

Our software prototype, *IslandViz*, was developed using *Unity3D*. Prior to island construction, a tool based on the work of Seider et al. [7], is used to extract all relevant information from the source code of the software. We tested our implementation on a desktop system with the following specifications: *Intel Xeon E5-2650*, 2.6GHz, 64GB Ram, *NVIDIA Geforce 1080*, *HTC Vive*. To validate the approach, a subset of the OSGi-based software project `RCE` (http://rcenvironment.de/) was visualized. It consists of 1700 classes, distributed over 500 packages in 160 bundles, along with 550 service and 1200 package dependencies. At all times, the achieved frame rate was well above 90Hz.

6 Conclusion and Future Work

We presented our approach for exploring OSGi-based software systems in virtual reality. We used an island metaphor to emphasize the modular aspects of OSGi and implemented an interaction technique to preserve user comfort, while inspecting large software systems. In future work, we would like to determine the practicability of this approach in aiding software comprehension tasks. Additionally, it would be very interesting to explore our visualization in AR.

References

[1] F. Fittkau, A. Krause, and W. Hasselbring. Exploring software cities in virtual reality. In *2015 IEEE 3rd Working Conference on Software Visualization (VISSOFT)*, pages 130–134, Sept 2015.

[2] A. Kuhn, P. Loretan, and O. Nierstrasz. Consistent layout for thematic software maps. In *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*, pages 209–218. IEEE, 2008.

[3] L. Merino, M. Ghafari, C. Anslow, and O. Nierstrasz. CityVR: Gameful software visualization. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 633–637. IEEE, 2017.

[4] J. Prothero, M. H Draper, T. Furness, D. Parker, and M. J Wells. The use of an independent visual background to reduce simulator side-effects. *Aviation, space, and environmental medicine*, 70:277–283, Apr. 1999.

[5] A. Schreiber and M. Brüggemann. Interactive visualization of software components with virtual reality headsets. In *2017 IEEE Working Conference on Software Visualization (VISSOFT)*.

[6] U. Schultheis, J. Jerald, F. Toledo, A. Yoganandan, and P. Mlyniec. Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3d tasks. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 117–124, March 2012.

[7] D. Seider, A. Schreiber, T. Marquardt, and M. Brüggemann. Visualizing modules and dependencies of osgi-based applications. In *Software Visualization (VISSOFT), 2016 IEEE Working Conference on*, pages 96–100. IEEE, 2016.

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

[9] C. Ware and G. Franck. Evaluating stereo and motion cues for visualizing information nets in three dimensions. *ACM Trans. Graph.*, 15(2):121–140, Apr. 1996.

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

[11] M. Yang and R. P. Biuk-Aghai. Enhanced hexagon-tiling algorithm for map-like information visualisation. In *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction*, VINCI '15, pages 137–142, New York, NY, USA, 2015. ACM.