2019 IEEE/ACM 1st International Workshop on Bots in Software Engineering (BotSE)

Adopting Conversational Interfaces for Exploring OSGi-based Software Architectures in Augmented Reality

Peter Seipel, Adrian Stock, Sivasurya Santhanam, Artur Baranowski,
Nico Hochgeschwender, and Andreas Schreiber
German Aerospace Center (DLR)
Simulation and Software Technology
Cologne, Germany
forename.surname@dlr.de

***Abstract*—We propose conversational interfaces as a convenient and complementary way for users to explore OSGi-based software architectures in immersive Augmented Reality (AR). By providing a conversational interface we aim to remedy some peculiarities of AR devices, but also enhancing the exploration task at hand. We exemplify a use case and sketch how different user utterances can be used to retrieve information about the to-be-explored OSGi-based software architecture. We identify crucial components such as natural language generation and intent recognition which are required to implement the user story and we outline the status of our implementation.**

I. Introduction

Exploring large-scale OSGi-based software architectures with interrelationships among multiple software components is a challenging exercise. To improve the exploration task, we developed multiple approaches in 2D, 3D, and Virtual Reality (VR) to visualize software architectures [5], [2]. To this end, different OSGi concepts such as *Bundles*, *Packages* and *Services* are mapped to elements of an island metaphor for visualization. For instance, bundles, packages and compilation units are represented as islands, regions and buildings (see Fig. 1). We demonstrated that such a visualization in AR is very appealing and helpful to, for example, get an project overview or to onboard new developers. However, for large-scale complex architectures exploration activities such as search and navigation among the elements remains a tedious and error-prone exercise. We observed that relying solely on gesture control makes exploration limited to navigation. For example, searching for a particular bundle by dragging the archipelago (see Fig. 1) around can be inefficient, when instead a query could be issued for finding the island.

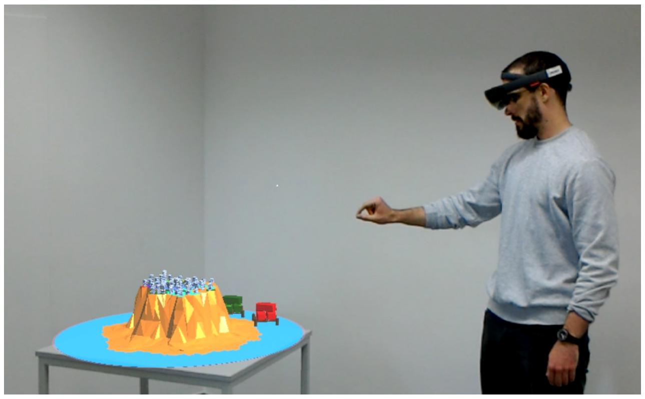
Fig. 1: Visualizing software architectures in Augmented Reality using the island metaphor.

II. Making the Case for Conversational Interfaces for Software Exploration

Thus, we aim to employ conversational interfaces as a convenient and complementary way for users to explore OSGi-based software architectures in AR with an additional modality. By utilizing several input modalities such as voice and gesture control provided by AR-devices, seamless navigation could be achieved. We argue that having both voice and gesture control could pave the way for advanced exploration activities such as simultaneously selecting architectural elements via gesture and requesting additional information about this element via speech. To exemplify both the opportunities and challenges in integrating conversational interfaces we discuss in the following paragraph a use case and derive potential intents, entities and additional service required for natural language processing. Table I shows some examples on how the conversational interface might work. The utterances form a possible sequence of commands a user can express. To process the first entry "*Please show me bundle core.auth*", the conversational interface needs to break down the significant information of the utterance. This can be realized by using a natural language unterstanding (NLU) service. This service is responsible for extracting intents and entities from a sentence. For the given utterance, the NLU service extracts the intent "*select_component*" and "*bundle*" and "*core.auth*" as the entities. Here, *bundle* is an OSGi-element and *core.auth* is the name of this element. Having those keywords should be sufficient to initiate a query which finds the corresponding bundle which then should be visualized as an island. To this end, a query generator is required which translates both intents and entities in suitable queries. This requires, that the to-be-explored OSGi architecture is represented in a queryable format. When processing the second utterance, "*Tell me more about this bundle*", the conversational interface again extracts the intent and the entities using NLU. However, the utterance itself doesnt provide all of the necessary information, as the user demanded information about *this* bundle. Using the current context, the conversational interface understands that *this* refers to the previously mentioned bundle "*core.auth*". Using this information, the interface again proceeds to build a query. The table shows more examples on how user utterances could be processed by the conversational interface. Note, that due to the use of the NLU Service, more than only one utterance can lead to the same outcome as long as the utterances have the same meaning. Therefore, the user doesn't have to know about specific phrasing to use the conversational interface. To this end, we see the biggest challenges in implementing the conversational interface lie both in training the NLU service and using and updating the context. The following section deals with the current implementation and sketches how the NLU service and the context handling are currently integrated.

III. Current Implementation

The current implementation is based on the Model-View-Control pattern. The view is represented by a VR/AR/MR device and the model is implemented by a database. This database contains the information about the OSGi software that is illustrated by the device. When receiving a speech input, the device uses a speech-to-text service and sends the result to the controller. To process this utterance, the controller uses the NLU service to extract the intent and entities as described in the preceding section. The controller then makes use of a *Intent-to-Query* service to receive a database query. This query is sent to the database and the controller receives the necessary information to fulfill the users demand. The controller sends these information back to the device. The device then reacts to the user's actual utterance, e.g. by focusing the demanded island. A context monitor keeps track of this process. This enables the conversational interface to handle utterances like "*what is this bundle about*" as explained above.

We use the *Microsoft HoloLens*[1] in our work. Being an IO device, it includes services such as speech-to-text, text-to-speech, gesture control and speakers. The HoloLens is just one of many VR/AR/MR devices; therefore our architecture is not restricted to this specific device. It is also possible to integrate our proposed system into other VR environments. When integrated with the conversational interfaces, the features of the HoloLens could be highly exploited to better facilitate exploration of architectures. The *Controller* includes an implementation of *Rasa core* [1] for the dialog management system. We use the graph database NEO4J in the repository, as graph databases are suitable choices for semantic queries, due to its nature of interconnected data. To convert the source code into a graph in NEO4J, we use the Open Source tool JQASSISTANT [3], [4]. The *NLU Service* in the *conversational services* uses RASA NLU [1] to detect intents and entities. The *Intent-to-Query* service builds Cypher queries with respect to intents and entities.

[1]https://www.microsoft.com/en-us/hololens

	Utterance	Intent	Entity	Extracted Information from current context
1.	"*Please show me bundle core.auth, select core.auth, ...*"	select_ component	bundle, core.auth	None
2.	"*Tell me more about this bundle, what is this bundle about?, ...*"	summarize_ information	bundle	core.auth
3.	"*Select the largest class of this bundle, show the largest class, ...*"	select_ component	class, biggest	core.auth
4.	"*How do I select an island?, ...*"	explain_ usage	None	AuthService Impl.java
5.	"*How many packages are inside this bundle?, ...*"	count_ component	bundle	AuthService Impl.java
6.	"*How many classes are inside this bundle?, ...*"	count_ component	class, bundle	AuthService Impl.java
7.	"*Which is the biggest package of this bundle?, ...*"	select_ component	biggest, package, bundle	AuthService Impl.java
8.	"*navigate to the smallest bundle?, ...*"	select_ component	smallest, bundle	de/rce/core/ component/

TABLE I: List of utterances and their intents and entities parsed.

References

[1] T. Bocklisch, J. Faulker, N. Pawlowski, and A. Nichol. Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*, 2017.

[2] M. Misiak, D. Seider, S. Zur, A. Fuhrmann, and A. Schreiber. Immersive exploration of osgi-based software systems in virtual reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, volume 00, pages 1–2, March 2018.

[3] R. Müller, D. Mahler, M. Hunger, J. Nerche, and M. Harrer. Towards an open source stack to create a unified data source for software analysis and visualization. In *2018 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 107–111, Sep. 2018.

[4] L. Nafeie and A. Schreiber. Visualization of software components and dependency graphs in virtual reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, VRST '18, pages 133:1–133:2, New York, NY, USA, 2018. ACM.

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