Combining Virtual and Robot Assistants—A Case Study about Integrating Amazon's Alexa as a Voice Interface in Robotics

Jan Jungbluth[1], Karsten Siedentopp[1], Rolf Krieger, Wolfgang Gerke, Peter Plapper.
[1]Both authors contributed equally to this work.

***Abstract*—Virtual assistants such as Alexa, Siri, Cortana and Google Assistant increasingly find ways to enter our homes and everyday lives, where they serve as powerful human-machine interfaces to control devices through natural language. In this work, we explore the use of virtual assistants to control single components within an intelligent industrial robot assistant system for disassembly applications. Following a short introduction and an overview of commercially available virtual assistants, we present our system architecture, which integrates Amazon's Alexa through an Echo Dot device. Utilizing the Alexa Skills Kit, we build a voice user interface that contains dialogs to control various device functions and assistive system behaviors. By connecting Alexa Voice Service with Amazon's Lambda and IoT web services, we can parametrize machine commands depending on the user's voice input and, through a Raspberry Pi, which routes messages between the internet and decoupled machine network, send them to the devices within the intelligent robot assistance system. With the reversed communication flow, we can update the current state of the devices in the Amazon Web Services IoT shadow, utilize the IoT Shadow in the Lambda functions, we are able to synthesize speech output with the Alexa Voice Services and vocalize this output through the Echo Dot to inform the user. One implemented assistive system behavior dialog is further explained in this work. In the conclusion, we address our positive and negative experience to date.**

I. Introduction

With recent progress in building intelligent robotic assistant systems [1] that cooperate or collaborate with humans in non-destructive disassembly processes, we encounter the problem of human-machine communication. In assistive robotic systems, communication is necessary to coordinate the actions of man and machine [2]. Coordination means (not exclusive) to define a common goal, assign tasks and actions as well as to synchronize them. From our performed experiments [3], we know that a graphical user interface with keyboard and mouse leads to flow-breaking, human-robot communication and, through that, to an insufficient human-robot interaction. To improve the human-robot communication, we have sought other communication modalities. Using a sensitive robot equipped with torque sensors on each axis, we can create a haptic communication interface, which can detect forces applied to the robot structure, such as a person pushing or pulling at its flange. This interface is currently used to synchronize the actions of man and machine. For example, while the user is executing his actions, the robot is waiting until the user pushes against the robot arm with the intention to confirm that the actions have been executed. Upon confirmation, the robot can begin to perform its actions. Since this approach is not always feasible, for example, when the robot structure is out of range of the user's arm, and the haptic information throughput is very limited, we engage alternative communication modalities such as natural language. In first experiments, an interface based on the .NET Speech Reconnection Library was used; tests revealed low speech recognition quality, especially in noisy environments. An alternative approach, the use of a virtual assistant (VA) as a voice user interface (VUI), was proposed, further investigated and implemented as a prototype. The technical implementation and first results are contend of this paper.

The further structure of this work is as follows: Chapter II defines a VA and provides a brief overview of commercially available VAs. Subsequently, Chapter III introduces the software architecture and explains the information flow through the single components. Thereafter, in Chapter IV we demonstrate the design of the dialogs upon the concrete assistive behaviors that we implemented. In conclusion, our experience is summarized in Chapter V.

Prof. Dr.-Ing. Peter Plapper is with the University of Luxembourg, Faculty of Science, Technology and Communication, Luxembourg, Luxembourg (e-mail: peter.plapper@uni.lu).

Prof. Dr.-Ing. Wolfgang Gerke is with the Trier University of Applied Sciences, Robotics and Control Department, Birkenfeld, Germany (e-mail: w.gerke@umwelt-campus.de).

Prof. Dr. Rolf Krieger is with the Trier University of Applied Sciences, Institute for Software Systems (ISS), Birkenfeld, Germany (e-mail: r.krieger@umwelt-campus.de).

Karsten Siedentopp is with the Trier University of Applied Sciences, Birkenfeld, Germany (e-mail: s17d54@umwelt-campus.de).

Jan Jungbluth is with the Company SEW-EURODRIVE GmbH & Co KG, Bruchsal, Germany (e-mail: jan.jungbluth.w@sew-eurodrive.de).

II. Commercially Available Virtual Assistants

VAs are described as the following:

A virtual assistant (VA) is a conversational, computer-generated character that simulates a conversation to deliver voice- or text-based information to a user via a Web, kiosk or mobile interface. A VA incorporates natural-language processing, dialogue control, domain knowledge and a visual appearance that changes according to the content and context of the dialogue.[4]

One of the first commercial VAs was Siri (Apple), which was released in 2011 and is preinstalled on all Apple smartphones. Other widely used assistants are Google Assistant, Amazon Alexa and Microsoft Cortana. These come preinstalled on a set of different devices: Cortana with the Windows operating system, Google Assistant on Android smartphones and Amazon Alexa with smart speakers.

The primary objective of a VA is to simplify human-computer communication and make human-device interaction more intuitive. All described VAs function similarly. When a request is performed on an assistant, it is packed as an audio file and sent to a server, where sophisticated algorithms for speech recognition and natural language understanding are embedded. The servers analyze the input and compute the probabilities that the input matches to each of the defined dialogs (i.e., utterances and words). Using a probability threshold, a decision of one correspondence is made and the defined related actions occur.

To control a device through a VA, three steps must be performed. First, the VUI with the objective's related dialogs (e.g., voice commands) must be created. The second step is to connect the device with the VA. In the last step, to control the device, the various user voice commands must be mapped to device actions. These three steps are performed in the system architecture as explained below.

III. Software and System Architecture

A. *Architecture Overview*

Within our application, we use Amazon Alexa embedded in the Amazon Echo Dot device. As mentioned above, the Echo Dot requires an active internet connection to connect to the Alexa Voice Service (AVS) [5] running on the Amazon Web Services (AWS) [6] ecosystem, shown in Figure 1. Within the AWS ecosystem, we further connect the AVS with the Amazon Lambda service (ALS) [7] and the ALS with Amazon's Internet of Things (IoT) [8] service. Furthermore, a microcomputer (Raspberry Pi) is used to route messages from the IoT cloud to a local network. Through the local network, the single devices of the intelligent robotic assistance system can be accessed to control or update the devices' state in the cloud.

Based on the information flow shown in Figure 1, we describe the various subjects in further detail.

B. *Alexa Voice Service*

The AVS combines several functionalities. First, it allows the user to develop, build, test and publish applications, called Alexa skills, by using the Alexa Skill Kit Developer (SDK) Console, a browser-based interface that is used to define the VUI of an Alexa skill. In the VUI, we define the voice commands (intents), and each intent is defined through different sentences (utterances), which activate the intents when spoken (i.e., the voice command is recognized). Furthermore, an utterance can contain placeholders for variables, called slots. Within a slot, we can define a fixed set of values that the variable can assume. We further explain the design of such skills in Chapter IV. The second purpose of the AVS is the functionality for speech recognition and natural language understanding, as well as access to these capabilities through APIs (i.e., the Alexa Skill Kit SDK). Based on the SDK, we developed a program to define the further actions that should occur when a specific intent is recognized. The developed program is written in Node.js and runs entirely within the cloud-based ALS.

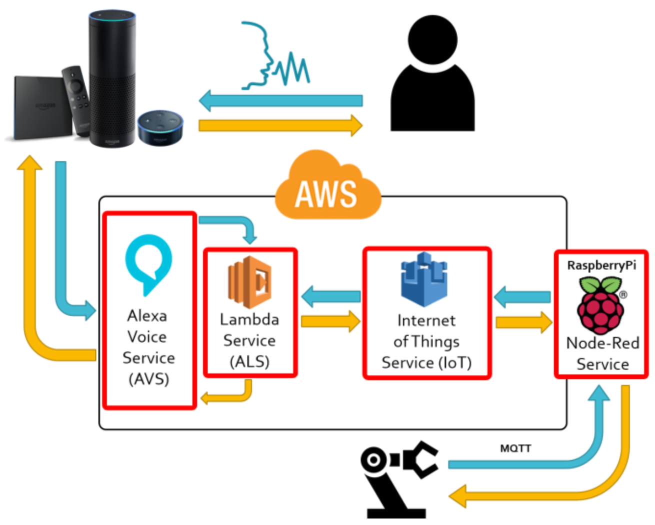

Figure 1: Architecture overview

C. *Amazon's Lambda Service*

The ALS is a serverless data processing service. This type of service dynamically adapts to the server workload by (de-)allocating more of a server's machine resources. This increases the scalability of the service regardless of the number of Alexa devices that send requests to the Alexa skill; the AWS cloud automatically scales performance, ensuring that the service is always reachable. Within the program, we run on the ALS, and we respond to user voice commands recognized by AVS. In the program, each intent is assigned to a machine command whose parameters are the slot variables. To send the machine command to the device, we use another service within AWS, the IoT service.

D. *Amazon's Internet of Things Service*

The IoT service allows for the integration of devices, so called IoT-things, with the AWS ecosystem. Amazon's IoT Device Management makes it possible to securely deploy, organize and monitor thousands of IoT devices. We use a Raspberry Pi as an IoT-thing to connect our system with the AWS. The Raspberry Pi is a single-board computer, developed by the British Raspberry Pi Foundation [9], which is a nearly complete computer the size of a credit card. In our work, we use the Raspbian operation system, a Debian-Linux distribution specially prepared and configured for the Raspberry Pi. Amazon provides an IoT device SDK for the Raspberry Pi that is necessary to communicate with the IoT. The Raspberry Pi runs a Node-Red [10] service, which is a programming tool that is used to wire together hardware devices, APIs and online services. It provides a browser-based flow editor to define the flow and manipulation of information between nodes (see Figure 2). With the IoT device SDK, custom nodes are implemented in Node-Red to use AWS. Through an encrypted virtual private network, we then send messages (using the MQTT protocol [11]) from the IoT service to our Node-Red service on the Raspberry Pi. Within Node-Red, we re-route the messages to the local network and the corresponding devices. Furthermore, we send messages about the system state back to the IoT service. The system state messages communicate the state of the devices and check whether the devices are connected. The state of the system is stored in the IoT service (within the so-called shadow) and called by the code in the ALS if the user requests the system state.

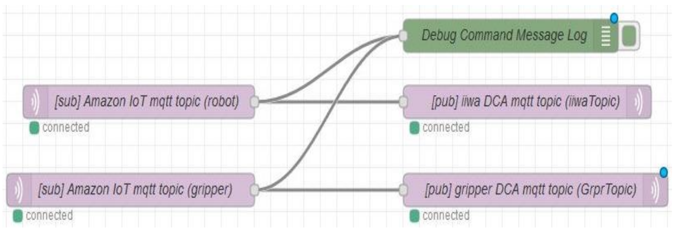
Debug Command Message Log
[sub] Amazon IoT mqtt topic (robot)
connected
[pub] iiwa DCA mqtt topic (iiwaTopic)
connected
[sub] Amazon IoT mqtt topic (gripper)
connected
[pub] gripper DCA mqtt topic (GrprTopic)
connected

Figure 2: A Node-Red flow with two MQTT subscriber nodes (left) connected to Amazon's IoT service, two MQTT publisher nodes (lower right) connected with the DCAs via the local network MQTT broker and one debug node that logs all messages

E. The Intelligent Industrial Robot Assistant System

We choose a distributed and multi-agent-based system design to establish our intelligent assistant system. Our system contains two types of agents: the device controlling agents (DCA) and an intelligent disassembly assistant agent (IDAA), shown in Figure 3. Within this architecture, we combine decentralized and centralized control approaches. Through the decentralized approach, each DCA can control another agent, and vice versa. The centralized control, through the IDAA, is necessary to coordinate and synchronize the actions of man and machine. As the name suggests, a DCA is a software that controls a device. Therefore, the DCA provides the device-specific communication and a unified communication interface, which is, in our case, the MQTT communication protocol. Depending on the controlled device and its purpose, we conduct further computation within the program.

In our application, one DCA controls an electric actuated two-finger gripper. Incoming machine commands from the unified communication interface are translated into the device commands and executed through the device-specific communication, shown in Figure 4. Simultaneously, the DCA monitors the gripper state (e.g., whether the gripper is open or closed) and publishes state changes on the unified communication interface.

Another DCA in our application controls an industrial robot (Kuka iiwa). This DCA receives commands via the unified commination interface and routes them through a TCP/IP connection to a robot program running on the robot controller. On the robot controller, a background-thread is listening for command messages on the in-going TCP/IP socket.

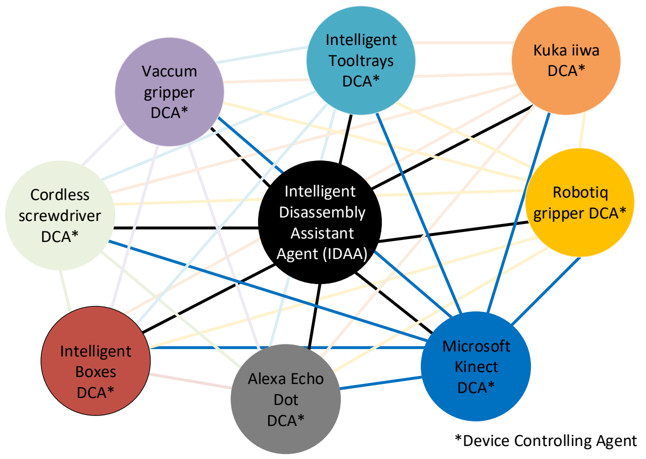
Vaccum gripper DCA*
Intelligent Tooltrays DCA*
Kuka iiwa DCA*
Cordless screwdriver DCA*
Intelligent Disassembly Assistant Agent (IDAA)
Robotiq gripper DCA*
Intelligent Boxes DCA*
Alexa Echo Dot DCA*
Microsoft Kinect DCA*
*Device Controlling Agent

Figure 3: The multi-agent system design

When a command message is received, the corresponding subprogram is executed in the main robot program. While executing a subprogram, we change the robot state accordingly. Through robot state changes, we transmit the new state through the second TCP/IP connection back to the DCA and publish the state on the unified communication interface.

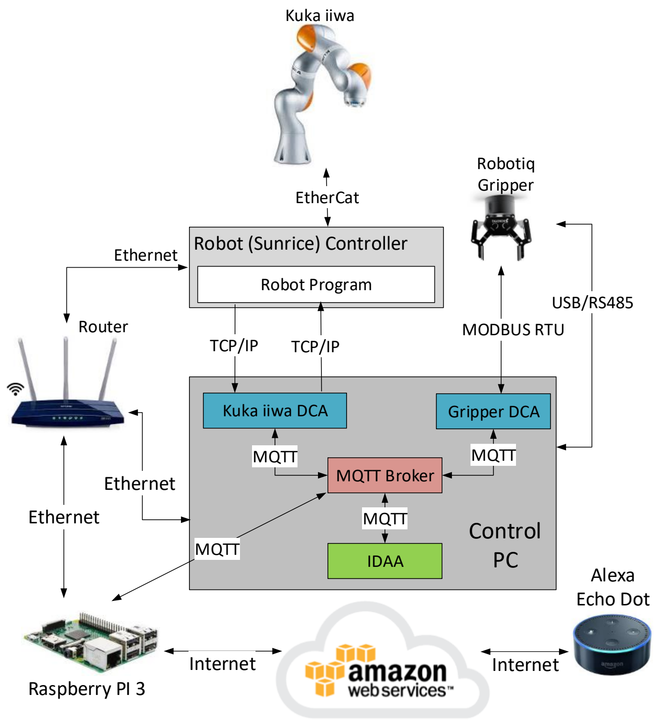

Figure 4: The technical implementation of the intelligent industrial robot assistant system (only the related gripper and robot DCAs are shown)

The second agent type is the IDAA, which creates a fluent and goal-orientated human-machine interaction that coordinates, controls and synchronizes the actions of the DCAs with human activities. The IDAA is capable of automatically determining the sequence of disassembly tasks upon a given goal description and a model of the product structure. Furthermore, the system provides different task-specific designed process models, which include different workflows. A workflow defines a specific distribution of labor between man and machine to complete a task. Upon the user-selected workflow and the current situation, the system can determine the necessary sequence of actions to perform the task as well as synchronize the action execution between the user and devices. Connecting the IDAA with the VUI is our next objective.

IV. THE VOICE USER INTERFACE

Within an interdisciplinary team, composed of computer scientists and experts from the field, the VUI was designed, implemented and tested. To structure the project, we relied on the design thinking process and focused on six tasks that the VUI should implement, shown in Table 1.

Table 1: Tasks for the VUI

1. Move the robot to its home position.
2. Move the robot to a position with a specific name.
3. Activate a specific waiting-mode on the robot.
4. Open and close the gripper.
5. The robot hands over a wrench of a certain size.
6. Get the current state of the assistance system.

For each of the six tasks, we designed intents with the browser-based AVS editor. Here, we explain the creation of one intent in more detail, describing the task of the robot handing over a user-requested wrench of a certain size. We named the intent HandWrench and defined the utterances from Table 2 to provide different sentences to recognize the related intent. If an utterance is recognized, the corresponding intent is activated. It is crucial to carefully consider what these utterances should be, since the VUI can only be used convincingly and intuitively if the sentences are well chosen and enough variants have been deposited.

Table 2: Utterances for the HandWrench intent

Give me the wrench, please.
Hand me the wrench.
Please hand me over the wrench size {wrenchSize}.
Give the wrench {wrenchSize}.
Give me the wrench size {wrechSize}.

Within some utterances in Table 2, the curly brackets ({ }) denote a slot. Within the slots, we can define the variable wrenchSize, which can assume different values. In our case, the wrenchSize variable can assume an integer value between 6 and 19, according to the available wrench sizes in our application. Because the wrench size is relevant information for executing the action, we define in the VUI intent that the wrenchSize value must be recognized.

The dialog shown in Figure 5 further explains the VUI. First, Alexa must be activated by means of a hot word: in this case, “Hey Alexa,” (Step 1 in Figure 5). The selection of the skill follows (Step 2); our skill name is “robot.” Next, the appropriate skill is started, and the user is greeted with the welcome message (Step 3). Subsequently, an intent can be executed; therefore, an utterance from the corresponding intent must be spoken (Step 4). Steps 1, 2 and 4 can also occur in one sentence, such as “Hey Alexa, ask Robot to give me the wrench.” Note that at this time, the necessary variable wrenchSize is not defined. This possibility is intercepted within our program running in the ALS. If the variable is not defined, we ask the user to specify the variable (Step 5). If the variable is set by the user (Step 6), the intent is completed (Step 7).

In our Lambda function, we now build up a command string, such as “HandWrench,Size12,” and send this over the IoT service to our Node-Red service running on the Raspberry Pi. Within Node-Red, we re-route the message to the corresponding DCA.

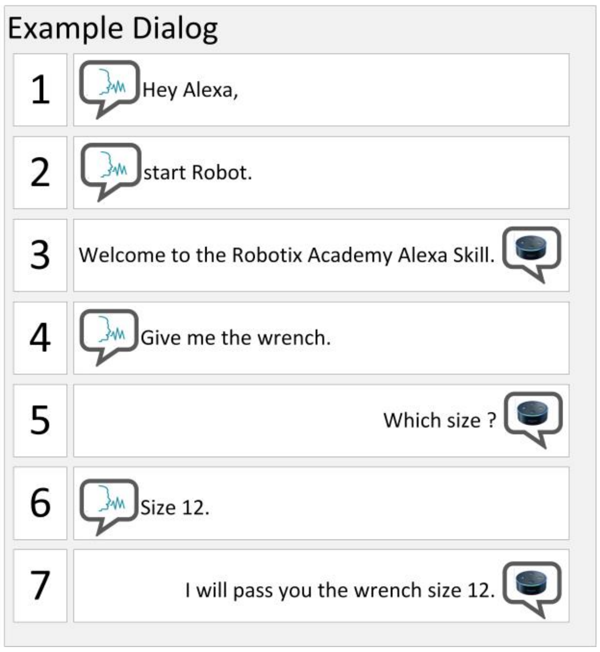

Figure 5: The complete dialog for starting our Alexa skill “robot” and the intent in which the robot hands over a wrench with a specific size

In this case, the corresponding DCA controls the robot and the corresponding robot subprogram HandWrench is executed. First, the robot commands the gripper DCA to move its fingers to a predefined position (to be open for the grasp action). Then, the robot moves first above and then exactly to the corresponding location of the size 12 wrench in the tool tray. Because the location of the wrench is fixed to the tool tray reference coordinate system, we must only teach the robot each wrench size’s position in the tool tray reference system once. If the tool tray is moved to a certain position, we must only determine the transformation from the robot reference system to the tool tray reference system. If the robot stands at the grasp position, the gripper DCA receives a command to grasp the wrench. Then, the robot re-tracks and moves to a predefined wrench handover position. At the handover position, the robot is waiting for a force applied to the robot structure. When the user pulls on the wrench, the robot recognizes the force and sends an “open” command to the gripper DCA. A video of this assistive behavior can be found in the “project of the month” video [12].

V. Conclusion

With the help of AVS, we were able to create a VUI with little effort; further, the voice recognition has proven its robustness, even in noisy environments. However, at some point of noisiness, the smart speakers will reach their limit and the use of headsets will be necessary. We have determined how useful this communication modality is in everyday (lab) work, and as human-robot communication and interaction become more natural, public visitors are increasingly likely to interact with the system. Further, we observe that visitors with industrial backgrounds enter deep thought after the demonstration and return with at least one possible application within their production. This suggests that applications exist in industrial environments. Getting a VUI into production was not feasible because industrial robots were not safe. But, with today’s (possibly) safe cobots (collaborative robots), this becomes feasible. However, disadvantages also exist. Sometimes Alexa misinterprets a sentence and begins unrelated speech, sometimes without any user request. This resembles speaking to a person on the phone. Without visual contact, a communication partner cannot always anticipate the cessation of speech and begins talking prematurely. Other disadvantages include that a steady internet connection to AWS is necessary and that the design of more complex dialogs is not yet supported.

With the usage of the ALS, we can run our code completely in the cloud and do not need to operate a dedicated server. However, developers must acknowledge the short lifespan of a Lambda function and that data cannot be stored in this service. The Alexa Skill Kit SDK used in our code greatly supports and leads to a moderate programming effort for our skill. However, debugging of such applications requires more effort than usual.

The usage of Amazon’s IoT service, especially the configuration and management of the connected devices, was the most time-consuming part of our work, but though the sophisticated rights and device management, we have a secure encrypted virtual private network to communicate with our Raspberry Pi. The only identified drawback is that the MQTT quality of service Level 2 is not supported; it is not possible to ensure that a message is delivered or received exactly once. Thus, it is possible that a device receives an action command multiple times if no further checks are implemented.

Further work will be related to connecting the voice interface with the IDAA to the control task as well as action specification, assignment, execution and synchronization.

References

[1] J. Jungbluth, W. Gerke, P. Plapper; An Intelligent Agent-Controlled and Robot-Based Disassembly Assistant, Conference Proceedings on the 2nd International Conference on Artificial Intelligence and Robotics (ICAIR), Prague, Czech, 2017

[2] W. Gerke, Technische Assistenzsysteme: Vom Industrieroboter zum Roboterassistenten. Berlin: De Gruyter Oldenbourg, 2015.

[3] S. Groß, W. Gerke, P. Plapper; Mensch-Roboter-Kollaboration für Demontageprozesse, Angewandten Automatisierungstechnik in Lehre und Entwicklung (AALE), Cologne, Germany, 2018

[4] Gartner's IT Glossary, https://www.gartner.com/it-glossary/virtual-assistant-va

[5] Amazons Alexa Voice Services Webpage, https://developer.amazon.com/de/alexa-voice-service

[6] Amazon Web Services Introduction, https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/introduction.html

[7] Amazon Lambda Service Documentation, https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/compute-services.html#aws-lambda

[8] Amazon IoT Services Documentation, https://docs.aws.amazon.com/aws-technical-content/latest/aws-overview/internet-of-things-services.html

[9] The Raspberry Pi Foundation, https://www.raspberrypi.org/

[10] Node-RED, https://nodered.org/

[11] MQTT Community, http://mqtt.org/

[12] Trier University of Applied Sciences (TUAS), Project of the month, https://www.umwelt-campus.de/ucb/index.php?id=12248