applied sciences

MDPI

Article

An Abstraction Layer Exploiting Voice Assistant Technologies for Effective Human—Robot Interaction

Ruben Alonso [1,†], Emanuele Concas [2,†] and Diego Reforgiato Recupero [1,2,*,†]

[1] ICT Division, R2M Solution s.r.l., 27100 Pavia, Italy; ruben.alonso@r2msolution.com
[2] Department of Mathematics and Computer Science and Human-Robot Interaction Lab, University of Cagliari, 09124 Cagliari, Italy; emanuele.concas@unica.it
* Correspondence: diego.reforgiato@unica.it
† These authors contributed equally to this work.

Abstract: A lot of people have neuromuscular problems that affect their lives leading them to lose an important degree of autonomy in their daily activities. When their disabilities do not involve speech disorders, robotic wheelchairs with voice assistant technologies may provide appropriate human–robot interaction for them. Given the wide improvement and diffusion of Google Assistant, Apple's Siri, Microsoft's Cortana, Amazon's Alexa, etc., such voice assistant technologies can be fully integrated and exploited in robotic wheelchairs to improve the quality of life of affected people. As such, in this paper, we propose an abstraction layer capable of providing appropriate human–robot interaction. It allows use of voice assistant tools that may trigger different kinds of applications for the interaction between the robot and the user. Furthermore, we propose a use case as a possible instance of the considered abstraction layer. Within the use case, we chose existing tools for each component of the proposed abstraction layer. For example, Google Assistant was employed as a voice assistant tool; its functions and APIs were leveraged for some of the applications we deployed. On top of the use case thus defined, we created several applications that we detail and discuss. The benefit of the resulting Human–Computer Interaction is therefore two-fold: on the one hand, the user may interact with any of the developed applications; on the other hand, the user can also rely on voice assistant tools to receive answers in the open domain when the statement of the user does not enable any of the applications of the robot. An evaluation of the presented instance was carried out using the Software Architecture Analysis Method, whereas the user experience was evaluated through ad-hoc questionnaires. Our proposed abstraction layer is general and can be instantiated on any robotic platform including robotic wheelchairs.

Keywords: robotic wheelchairs; cloud robotics; language understanding; human–robot dialogue; voice assistant technology

Citation: Alonso, R.; Concas, E.; Reforgiato Recupero, D. An Abstraction Layer Exploiting Voice Assistant Technologies for Effective Human—Robot Interaction. *Appl. Sci.* **2021**, *11*, 9165. https://doi.org/10.3390/app11199165

Academic Editors: Ana Lopes and Alessandro Di Nuovo

Received: 27 July 2021
Accepted: 27 September 2021
Published: 2 October 2021

1. Introduction

People with neuromuscular problems tend to lose a significant degree of autonomy in their daily life. To increase their mobility and independence, power wheelchairs may provide a valuable solution. Due to the deterioration in their physical ability, some of them become unable to use power wheelchairs. A potential solution is represented by robotic wheelchairs, which may provide appropriate human–robot interaction to people with severe motor impairments. They may include intelligent assistive navigation systems to help people according to their capabilities. This reflects the trend of the last years, where the employment of robots has changed from common usage within big manufacturing industries to locations such as people's homes, hospitals, schools, hotels, restaurants, where the daily-life is carried out. As already happened several years ago with the diffusion of smartphones, we are assisting nowadays a similar wide spreading of the employment of robots for different tasks that will contribute to the development of new architectures, frameworks, and applications within the robotic domain. This happens because research

in different Information and Communications Technology (ICT) areas has brought about several cutting-edge technologies that have shaped new robotic platforms. These platforms can be improved and extended into several automatic robotic applications to help people in several domains (e.g., health, education, engineering, entertainment).

Robots have also been used as socially assistive entities playing an important role for children with autism spectrum disorders [1] and the active aging of older adults [2]. Active aging is a current topic and has an enormous economic and social impact. According to [3], in 2017, the EU-28's total health-care spending was EUR 1.73 trillion. This accounted for around 10% of the EU GDP (Gross Domestic Product) where curative care and rehabilitative care services incurred on more than 50% of healthcare expenditure in the majority of European Union States. The total spending is expected to increase to 30% of GDP by 2060 [4], especially due to the increase in the aging population. In particular, reduced functional independence and loneliness, unavoidable in older people, who lose connection with their friendship networks and for whom it is more complex to start new ones, can further increase isolation and depression leading to a further increase in healthcare costs [5].

Nowadays, it is possible to program robots to the assistance and rehabilitation of people, and this improves the efficiency of the treatments and may alleviate care centers' workloads, leading to a reduction in the costs for both the care centers and the patients [6], reducing public and private long-term care and health-care expenditure. Furthermore, patients might even be treated in their homes, remotely, through telerehabilitation, thus reducing the inconvenience they experience.

Moreover, robots are being constantly and widely used in education [7] and STEM (Science, Technology, Engineering and Mathematics) in general [8], for example to allow people to practice languages. One example of a robot used within education is Elias [9], a social robot that helps students learning foreign languages. Another one is the Makeblocks' mBot [10], which is able to teach kids to code.

To further engage users, a robot must be equipped with technology that can understand what the user is saying, and identify the hidden psychological traits like emotions, sarcasm and irony [11].

To tackle this issue, the improvement of voice assistant technology has enabled several changes in life today. Google Assistant [12], Apple's Siri [13], Microsoft's Cortana [14], Amazon's Alexa [15], Wit.ai [16] and Snips.ai [17], an open source and privacy oriented solution, are all well-known examples of digital voice assistants available on the market today. Such technologies can be used to access apps, services, and software, reading out the daily news, setting timers, or tapping into music playlists, and control IoT products and sensors. As each voice assistant makes available its own APIs for developers, it is possible to integrate their technology for several purposes and in different devices, robots included, bringing about new business opportunities in diverse areas [18]. They can also be employed within robotic applications. In fact, they help the robots maximize engagement with the users by interacting with the surrounding world, for example by recognizing an object, by executing particular action commands or by helping the user control the domestic environment (e.g., smart homes).

A robotic wheelchair consists of a semi-autonomous controller with at least two agents, the user and the robotic wheelchair. They collaborate with each other during the navigation process. When speech disorders such as dysarthria or apraxia are not involved, voice assistant technology might increase the effectiveness of robotic wheelchairs, as the latter must recognize human intent in every step of the navigation process and make the right decisions. In recent years, several robotic wheelchair prototypes have been developed with the aim of improving the mobility capabilities of severely motor-disabled people [19].

In line with all the aforementioned issues, in this paper we introduce an abstraction layer for Human–Robot Interaction (HRI) that employs voice assistant technologies. Such a design allows the robots to increase their interaction with the users by exploiting the power of recent voice assistant technologies. The proposed abstraction layer can be employed for any robotic platform including robotic wheelchairs.

In fact, one of its main advantages is that it is possible to embed and integrate any external software framework needing high computational power and that can reside in cloud systems or external servers, leaving the robot's resources free and focused for the interaction with the users, which therefore results in more fluid and efficient interactions. We have also instantiated a use case out of the proposed general abstraction layer by choosing all the components with off-the-shelf tools and using an extended version of the humanoid NAO robot as the robotic platform to test its flexibility towards other kinds of robots. The proposed abstraction layer does not assume any technical choice but is a high-level design and can be instantiated as well in any robotic platform. The instance we developed is one possible way to implement the proposed design.

Therefore, the contributions of our paper are the following:

- We propose an abstraction layer for HRI integrated with voice assistant technologies;
- Out of the proposed abstraction layer, we instantiated a use case;
- For the proposed use case, we show how to integrate voice assistant technologies (we used Google Assistant and its developer's suite) with applications written for the used robotic platform; as such, we developed a Chess game by using Dialogflow [20] of Google that we integrated within the proposed use case;
- We illustrate the kinds of applications that can be developed for the proposed use case using and combining cloud or server platforms, DialogFlow, and Choregraphe, therefore leaving free the robotic platform's resources;
- We identified six different stakeholders and five scenarios to evaluate the proposed use case using the Software Architecture Analysis Method (SAAM) and used questionnaires to test the related user experience.

The remainder of our paper is organized as follows. In Section 2 we report some related work regarding robotic wheelchairs and voice assistant technologies. In Section 3 we introduce the overall abstraction layer and what type of applications can be included and integrated. The instance we developed is illustrated in Section 4 where we also explain what is needed to create a new application. The applications that we developed and integrated into our use case are detailed in Section 4.1. An evaluation of the use case is conducted in Section 5. Finally, Section 6 ends the paper with conclusions, ideas, and directions in which we are headed.

2. Related Work

In this section, we will discuss works related to the robotic wheelchair domain and Voice Assistant tools.

2.1. Robotic Wheelchairs

There are different works in the literature where robotic wheelchairs have been developed with different designs and tasks. Miller and Slack [21] were the first ones to propose two low-cost robotic wheelchair prototypes to assist the user in avoiding obstacles, reaching a fixed destination, and maneuvering through doorways and other narrow or crowded areas. Authors in [22] developed a robotic wheelchair trainer that automatically follows a path highlighted by a line on the floor using computer vision, haptically guiding the driver's hand in appropriate steering motions using a force feedback joystick. Moreover, the work presents an evaluation analysis where the chair was used to teach children without motor impairment aged 4–9 to drive the wheelchair in a single training session to assess its learning simplicity. Other authors presented the development of a intelligent robotic wheelchair focusing on its mobility assistance design [23]. More in detail, they designed a user interface for older adults with declining abilities in perception, motor control and cognition. The device consists of a moving vehicle, a multiple degree-of-freedom seat adjustment mechanism, and an information/communication module. Its aim is to facilitate physical interaction with the environment and information exchange and interpersonal communication with the outside world. Other authors presented the robotic wheelchair MAid (Mobility Aid for Elderly and Disabled People) [24]. It was built on top

of a commercial wheelchair extended with an intelligent control and navigation system. Among its tasks, it includes navigation in narrow, cluttered environments and through wide, crowded areas. Therefore, it has been designed with the main purpose to support and transport people with limited motion skills. Some researcher focused their studies on the navigation methods for autonomous wheelchair robots. This is the case of authors who in [25] presented a new qualitative navigation system for typical home environments. The proposed approach receives as input a line diagram of the robot environment and converts it into an enhanced grid where qualitative representations of variations in sensor behavior between adjacent regions in space are stored. These representations are fed to an off-line planner which stores appropriate motion commands at each grid cell that will ideally move the wheelchair in and out of each room in a typical home environment. This input, the starting and destination positions of the robot, are fed to an online controller which compares the actual behavior of the sensors with the one stored in the grid and estimates the current position of the robot in order to retrieve the planner instructions and to combine these instructions with appropriate risk avoidance behaviors during navigation. Another work that focused on the design of robotic wheelchairs for autonomous path planning and high performance computing for real-time data processing has been proposed in [26] within the medical domain. Results proved that the proposed device was efficient in detecting obstacles. Further contributions of researchers made use of different innovative technologies that kept extending current power wheelchairs with new capabilities. The term "smart wheelchair" has thus been coined to indicate a power wheelchair to which computers, sensors, and assistive technology are attached. The work described in [27] aims to provide a complete state-of-the-art overview of smart wheelchair research trends starting from power wheelchairs and the innovations that have gradually been leveraged to transform them in smart wheelchairs. Another survey with a comprehensive literature review of smart wheelchairs and future research in the field of artificial intelligence, sensor technologies, and robotics has been presented by authors in [28]. Authors distinguished each smart wheelchair according to their form factor, their input methods, the sensors they had, their control software, their operating modes, their internal mappings and landmarks, and their commercialization.

Several research projects within the robotic wheelchair domain have been funded. For example, RADHAR (https://cordis.europa.eu/project/id/248873 (accessed on 30 June 2021)) proposed a framework to estimate the trajectory the robot should execute based on driver behavior and pervasive environment perception. Then this information was used for safe navigation with a level of autonomy depending on the user's capabilities. AUTONOMY (https://cordis.europa.eu/project/id/977/es (accessed on 30 June 2021)) was another project funded by the European Commission whose aim was to provide conventional powered wheelchairs with the functionality required for autonomous navigation adapted to the requirements of specific users. One more project funded by the European Commission is FreeWheel (https://cordis.europa.eu/project/id/768908 (accessed on 30 June 2021)), which promotes social inclusion of disabled and elderly people through a urban mobility solution consisting of a unit integrating an autonomous "smart active" module, multiple custom interfaces and an app. The rationale is to allow customization, both on the user and on the vehicle side, through the implementation of a modular concept based on standard reconfigurable, low-cost modules (e.g., engine, gears, control unit, HMI, etc.) and on ultra-customized interfaces (e.g., body-to-vehicle, engine-to-vehicle, vehicle-to-infrastructure, etc.), produced via additive manufacturing.

2.2. Voice Assistant Tools

The current spread of Voice Assistant Tools is greatly influenced by their adoption in private life, where they are mainly used in elderly care, domotics, and entertainment [29]. As far as elderly care is concerned, authors in [30,31] proposed a method that employs a voice assistant in a health smart home environment to provide health-care services for the elderly who want to continue living in their homes and who often perceive smart home

objects as way too complex to use. Similarly, voice assistants are being used to ease access to online services with speech-centric multimodal interaction for elders [32]. Concerning the domotic domain, authors in [33] introduced an ambient intelligence controller, which allowed people to control all their domestic appliances by speaking in natural language to a voice assistant named Maior-Domo. As far as entertainment is concerned, authors in [34] designed a voice assistant for in-car children's entertainment, named PANDA, where activities were started with simple vocal commands like "show movie" or "play game", and where games were designed to be played only with vocal commands.

Furthermore, authors in [35] developed an end-to-end personal assistant named Lucida (also known as Sirius), which works with both speech and images. They observed that the creation of voice assistants is expensive both in economic and computational terms. The reason was that they involved complex methods of natural language processing to understand and process users' speech. In their work, they describe the elements a voice assistant consists of and how they interact with each other, explaining the algorithms that were used and how a data center should be designed to handle their computations. Moreover, they provide a real system analysis of Lucida to discover bottlenecks, describing the methodology and the platform used to speed up the most important components. The idea of decentralizing services in independent domain expert systems goes against other works such as [36], where language understanding and dialog components are centralized to provide additional cross-domain flexibility and re-use. Lastly, authors in [37] discussed a less scaled architecture, listing requirements and use cases of a voice assistant and presenting a format to describe intents which are used to communicate with the server through the MQTT protocol [38].

3. The Proposed Abstraction Layer

In this section, we describe the general abstraction layer we propose in this paper. It consists of four main components, detailed in the following subsections and shown in Figure 1.

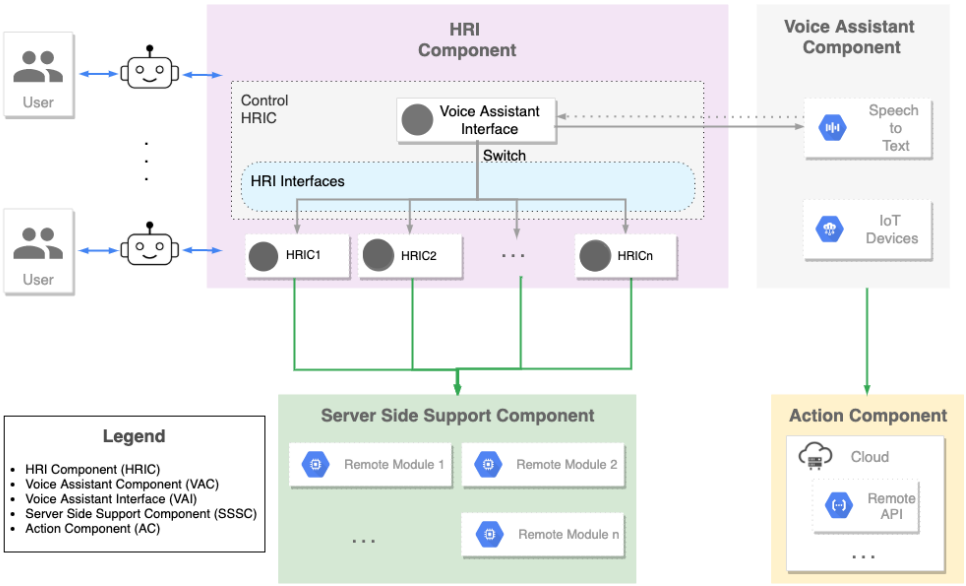

Figure 1. Proposed abstraction layer. The HRIC interacts with the user and sends the user's audio to the VAC. The VAC decides which modules should be triggered according to the input audio. It sends to the HRIC either a string indicating which HRIC needs to be called or the result of the voice assistant cloud (if the user asked something in the open domain). Every application is generally composed of one HRIC, uploaded into the robot, which can have an associated SSSC, residing in the cloud and that can be shared among all the robots. Actions can work without an associated HRIC. Several HRIC, SSSC, and Actions can be added.

3.1. HRI Component (HRIC)

The HRI Component (HRIC) is responsible for the interaction of the robot with the users. As such, several HRICs are usually loaded into the robot, control its behaviors, and have access to its sensors and parts. They are usually lightweight programs because, usually, the robotic platform does not have many hardware resources available for extra-computation. A certain robotic application may be developed entirely through an HRIC if it does not require heavy computational power or huge storage capabilities.

There should always be a *control HRIC* that acts as a master and decides which other HRIC to run, depending on which command the user said (parsed by the Voice Assistant Component).

Three mandatory elements of the control HRIC are the following:

1. A voice assistant interface (VAI): this is an abstraction of the embedded voice assistant tool being used. It includes the software elements to receive the audio, analyze it, send it to the voice assistant cloud, and retrieve the answer. If an HRIC has been invoked (using a particular voice command), then a string is sent to the next element without forwarding the voice command to the voice assistant (i.e., the framework responsible for the control of HRICs takes over and internally handles the commands without forwarding to the voice assistant).
2. Switch application: when the user invokes an existing HRIC (usually by calling the HRIC to invoke), the string returned from the previous application is forwarded to a switch element that routes the request and enables the called HRIC.
3. HRIC interfaces: these are connected to the switch element and are linked to the HRIC that performs specific actions. They can be removed or extended as soon as new HRICs are included in the design. When the called HRIC ends its execution, the control is returned back to the voice assistant interface.

To note that multiple different HRICs may be combined so that the robotic platform can be executed to run more than one actioncon currently. This may be done by either creating one HRIC which includes different existing HRICs or by the VAI if the robotic platform supports that.

3.2. Voice Assistant Component (VAC)

The VAC is a key component of our abstraction layer. It receives the audio from the user in streaming from the control HRIC and reacts in different ways. It is the responsibility of the VAC to detect the end of the text the user is expressing. The received audio is first processed through a speech-to-text engine. If the text matches the starting command of any of the applications loaded into the robot, then the control HRIC activates them. Otherwise, the voice assistant is enabled for the interaction with the user. In this way, the user can either play with any of the applications loaded into the robot or interact with the robot in the open domain to ask anything he/she desires. The VAC can employ any of the existing state-of-the-art voice assistants tools such as Google Assistant, Amazon Alexa, Apple Siri, Microsoft Cortana, Wit.Ai, and so on. In general, any voice assistant tool with the following requirements may be employed for the VAC:

- It is possible to send the user's audio by using the SDK of the adopted voice assistant tool to its related cloud server handling the APIs for voice recognition;
- The answers to the user's questions are returned in a text or audio format. If the former occurs, the robot voice (or text-to-speech tools if the robotic platform is not equipped with speakers) could be used to read the answer. If the latter, the audio is simply played;
- Custom script commands can be used to extend the voice assistant tools (e.g., Google Assistant's Custom Device Actions if Google Assistant is employed). This would allow triggering of other behaviors and not forwarding the request to the voice assistant tool.

3.3. Server Side Support Component (SSSC)

Following the rationale of cloud robotics, which is having robots benefiting from the power computation, storage, and communication resources of the modern data centers in the cloud, in our abstraction layer we have introduced the Server Side Support Component (SSSC). It consists of a cloud or servers that store the data and algorithms required for a particular task demanding high computational power or high storage capabilities. The HRICs can communicate with related algorithms stored within the SSSC through REST APIs. They can be called via HTTP requests by the HRICs directly running into the robot. Every SSSC should have an associated HRIC to exchange data. Moreover, each SSSC can run independently from each other and the other components. We can have as many SSSC as desired.

Therefore, all the HRICs that require machine learning approaches, big data computation, cognitive computing capabilities, and so on, will need a certain number of SSSCs for the heavy computation. In such a way, all the cutting-edge discoveries in fields such as Computer Vision, Artificial Intelligence, Big Data, Semantic Web, Natural Language can be employed for an effective HRI.

This model is flexible and scalable because:

1. The same SSSC can be used by different robotic platforms and different HRICs at the same time;
2. The HRICs, to interact with SSSCs, need to be connected to the Internet and perform HTTP requests, which are very simple and not CPU-consuming tasks. For this reason, it is possible to upload a large number of HRIC before ending the physical space of a given robotic platform;
3. An unlimited number of SSSCs can be uploaded in the cloud or on external servers.

3.4. Action Component (AC)

Actions extend the functionality of the VAC and are available for a large amount of state-of-the-art voice assistant tools. These actions allow developers to increase the effectiveness of the HRI by creating a personalized interaction for users, allowing things to be done with a conversational interface that ranges from a rapid command to turn lights on to a longer conversation, like playing a chess game. In more general terms, an action defines the behavior for a specific user intent and the corresponding fulfillment.

Unlike SSSCs, ACs do not need to have an associated HRIC (unless they need access to robot parts or sensors different than robot's speakers and microphones), because actions reside in the voice assistant cloud and they can be triggered directly from the VAC by giving explicit commands to the voice assistant.

All the actions created for voice assistant tools can be used out-of-the-box with any robotic platform, allowing the robot to have an entire set of applications ready to use. This leads to (i) a very flexible design because actions can also be reused with any device connected to the assistant platform; (ii) further scalability as an undefined number of actions can be created and uploaded.

4. The Proposed Use Case

In this section, we propose a use case implementing the abstraction layer shown in Section 3. As a robotic platform, we have chose NAO (https://www.softbankrobotics.com/emea/en/nao/ (accessed on 30 June 2021)). To develop the HRIC we adopted the Choregraphe suite. Note that any robotic wheelchairs can be employed for different use cases. The VAI of the Control HRIC is a peculiar interface used to communicate with the voice assistant platform. It includes a thread that records the audio coming from the robot's microphone using the command-line sound recorder for ALSA (http://alsa-project.org/ (accessed on 30 June 2021)) drivers with the *arecord* command. Its output is streamed to a middle-ware layer that performs buffering and sends the audio to the voice assistant platform. Through the middle-ware, the VAI receives and parses the response sent by the voice assistant platform. An example of a response may be the name of the application

to run. The switch is a simple module that routes the VAI output and starts the desired application of the user.

For the VAC module, we chose to employ Google Assistant, which fulfills all the requirements mentioned in Section 3.2 (accessed on 30 June 2021). The Assistant Embedded APIs (https://developers.google.com/assistant/sdk/reference/rpc/google.assistant.embedded.v1alpha2) are used to send the audio to the assistant and return a response to the user. Examples are answers to simple questions (e.g., "What time is it?") or interactions with a Google Action. Google Assistant can be extended with Custom Device Actions (https://developers.google.com/assistant/sdk/guides/library/python/extend/custom-actions (accessed on 30 June 2021)), which allow the robot to have special abilities not covered by the default Google Assistant's traits (https://developers.google.com/assistant/sdk/reference/traits/ (accessed on 30 June 2021)).

ACs are built with Actions on Google (https://developers.google.com/actions/ (accessed on 30 June 2021)) and were used to develop one application called *Mr. Chess*, which allows a user to play chess with voice commands. Actions On Google work in collaboration with Dialogflow, a user-friendly tool that allows the use of machine learning to understand the natural language of users.

In our use case, we developed six applications. Four of them use a dedicated SSSC to perform heavy computations with an associated HRIC interface for handling the interaction with the user. The interfaces can communicate with SSSCs through REST APIs. One of them consists of an HRIC only without the support of SSSC. Another is entirely developed using the AC component.

4.1. The Developed Applications

The applications we included in our use case represent different combinations that can be developed on top. The first one, the Bingo game, is a simple HRIC without any SSSC or AC. Then, we present four modules with a similar scheme, consisting of HRIC and SSSC. Finally, we describe the Chess game we developed through the AC only.

4.1.1. Bingo Game

This application consists of a simple HRIC created to play Bingo with one or more persons. It works without any external SSSC. Thus the HRIC is responsible for everything: the interaction with humans and the game logic. Applications such as this might be very entertaining for the elderly. The Bingo Game starts as soon as the user says to the robot *Hey NAO, let's play Bingo now*.

Bingo HRIC

When the application starts, NAO explains the rules of the Bingo game. It then waits for an acknowledgement from the user to start the game. Once started, the robot keeps extracting numbers until the user says a stop word, such as *bingo*, *line*, *repeat* or *stop*. If the user says *bingo* or *line*, the robot asks the user to say the extracted numbers (for the line or the bingo). After the user response, the robot double checks the numbers with the user and, if the user confirms, checks them among those previously extracted. In case of winning the robot performs an happy dance otherwise it performs a disappointing animation. In any moment of the game the user may say *repeat* to ask the robot to repeat the last extracted number, *faster* or *slower* to increase/decrease the speed of extraction and *exit*, to exit and close the application giving control back to the VAC.

4.1.2. Semantic Sentiment Analysis Application

The Semantic Sentiment Analysis Application allows NAO to understand the polarity of the user's input sentence. The application consists of two parts: the HRIC and the SSSC. The HRIC, enabled by the VAC after the user's expression *NAO, play sentiment analysis*, handles the interaction between the robot and users: NAO waits for a natural language expression, which is then converted into text by a speech recognition module

powered by Nuance (https://www.nuance.com/ (accessed on 30 June 2021)) and the converted text is sent, via REST APIs, to the associated SSSC for the classification task. The SSSC includes the Semantic Sentiment Analysis engine [39–46], which employs a Deep Learning approach using Recurrent Neural Networks (further details on the used deep neural network, training, test data, and tasks can be found in [47]). The SSSC, then, returns an output class, *positive*, *neutral* or *negative*, and the robot performs a different animation based on that. NAO keeps waiting for input sentences to identify as positive, neutral or negative until the user says *exit* to end the Semantic Sentiment Analysis application and returns the control to the VAC.

4.1.3. Generative Conversational Agent Application

The Generative Conversational Agent Application included in the use case is capable of holding an open-dialog conversation with the user. As a result, NAO is thus able to reply with coherent and sensible responses and according to the user's precedent utterances.

Once the application is enabled by the VAC (when the user says *NAO, let's talk*), the robot waits for the user to talk. Similar to the previous application, this one consists of an HRIC which handles the interaction with the user, and an SSSC, where the server runs the generative conversational agent and provides REST APIs. After the speech-to-text, the input text is sent to the SSSC, which uses an embedding layer based on the Stanford GloVe word embedding project (https://nlp.stanford.edu/projects/glove/ (accessed on 30 June 2021)) to generate an answer. Further details on the adopted generative conversational agent can be found here [39].

4.1.4. Robot Action Commands Application

The Robot Action Commands Application allows NAO to execute multiple natural language action commands spoken by the user. Commands such as *NAO, raise your left arm and sit down* or *NAO, walk forward raising your arm* can be given through this application. This application exploits an ontology that is fed to a natural language processing engine that has been defined in this paper. The architecture of this application consists of an HRIC and a corresponding SSSC. As soon as the application is enabled by the VAC, the HRIC sends input commands of the user to the SSSC for processing.

The output of the SSSC is a recognized action to be sent back to the HRIC, which will instruct the robot to perform the related physical action. If the action is incomplete, the robot asks the user for the missing information. This occurs when a user misses some entity, for example in *NAO, please raise your arm*. In such a case, the Natural Language Processing engine that has been developed within the SSSC knows that there might be two possible actions, related to the two arms of the robot. The HRIC then is informed of the two choices and interacts with the user asking to specify further. Moreover, the HRIC has two modes, *STATEFUL* and *STATELESS*. When the former is set, all the positions of the robot between different commands are maintained. When the latter is set, the robot goes back to its default position after it performs each requested action. Further details about the NLP engine, the ontology, and how they interact with each other to identify robot actions can be found in [48].

4.1.5. Object Detection Application

The Object Detection Application is enabled by the VAC when the user says to the robot *NAO, execute object detection*. The HRIC is therefore activated, and the robot informs the user. When he/she says *recognize*, the robot will take a picture using its front camera. The photo will be sent to the associated SSSC, which contains the object detection server employing the TensorFlow Object Detection API (https://github.com/tensorflow/models/tree/master/research/object_detection (accessed on 30 June 2021)) used to compute the bounding boxes and the related object categories with the associated confidence values. The SSSC can recognize 91 different types of objects, and for two particular object categories, dogs and cats, it can perform a fine-grained classification and recognize their breed.

After the HRIC sends the picture to the SSSC, it returns the output class of the recognized object to the HRIC. The HRIC then asks the user if that object is correct. If yes, the new annotation is saved together with the picture and will augment the training set. The model is periodically re-trained with the new data. Further details on how the object detection server works can be found in [49].

4.1.6. Mr. Chess Application

Mr. Chess is a Google Assistant action conceived to play chess with a user. This application demonstrates how standalone AC applications fit into our architecture. NAO acts like middleware between the user and the Google Assistant action. For this reason, the HRIC is not responsible for managing the interaction with humans, and the audio recorded by NAO's microphone is sent to Google Assistant, where it is elaborated and converted into text. The correct intent is found by the Dialogflow Agent, which recognizes a specified pattern.

The Chess Application is composed of two elements: a Dialogflow Agent and REST APIs.

Dialogflow Agent

The Dialogflow Agent consists of two main components: intents and entities. They both manage and route user's requests. Intents are used to map users' input to responses, whereas entities are used to extract and understand useful data from intents inputs. The Mr. Chess application consists of three main intents:

- *Default Welcome Intent* is the default intent of the dialog. It starts when the user begins the conversation, and it first explains the rules of the game. At the first run, it asks for confirmation to extract the user's name from the Google account used in the callback function triggered by the intent. If it is a replay, the robot greets the user. Next, the intent checks if there is a pending active game, and the user is asked if either he/she wants to re-join it or if he/she wants to create a new one.
- *Make a Move* manages most of the interaction between the action and the user. It is capable of understanding all the possible chess moves, which will be sent to the Chess REST APIs through an HTTP call. According to the answer of the REST APIs, the action will answer accordingly. If, for example, the move is invalid, the user will be informed. The game continues until either a stalemate or a checkmate occurs. The *Make a Move* intent can be seen in Figure 2;
- *See Board* is a helper whose purpose is to print the updated chessboard when the user wants. Note that this function works when the used robotic platform has a device to show output. As such, it does not work in our test-case.

A new entity type, named *chess_coordinates*, was created. It specifies all the possible chess moves that can be found in users' utterances. An example of its definition is shown in Figure 3.

Chess REST API

The chess REST API is built on top of Chess.js (https://github.com/jhlywa/chess.js (accessed on 30 June 2021)) and ChessCorp Artificial Intelligence: Kong (https://www.npmjs.com/package/chess-ai-kong (accessed on 30 June 2021)). Chess.js is a Javascript library used for chess move verification and validation, for the identification of specific situations (e.g., stalemate, check and checkmate) and chess piece placement. The latter is an artificially intelligent algorithm search engine, based on Alpha Beta Pruning (https://en.wikipedia.org/wiki/Alpha%E2%80%93beta_pruning (accessed on 30 June 2021)). The Chess API is written in NodeJS using the Express framework (https://expressjs.com/ (accessed on 30 June 2021)). For every user command (new game, load game, move, etc.), an HTTP call is available. The states of games is stored in a NoSQL database, MongoDB, hosted in the cloud server MongoDB Atlas. The code of the Chess Application is freely accessible (https://github.com/conema/chess-api (accessed on 30 June 2021)).

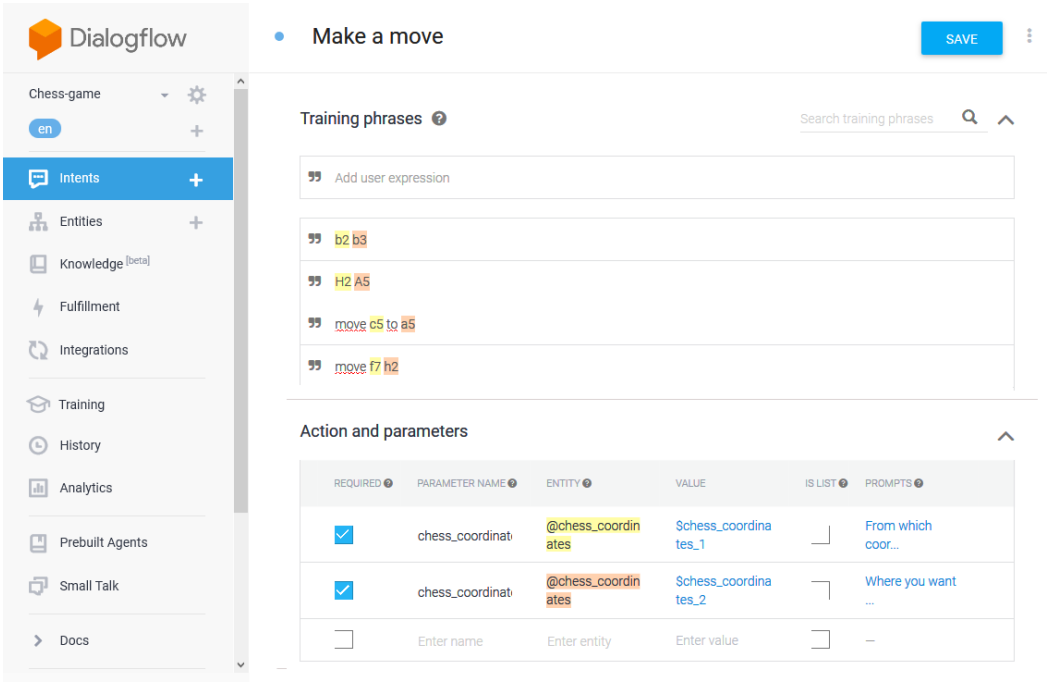

Figure 2. *Make a move* intent from Mr. Chess Dialogflow console.

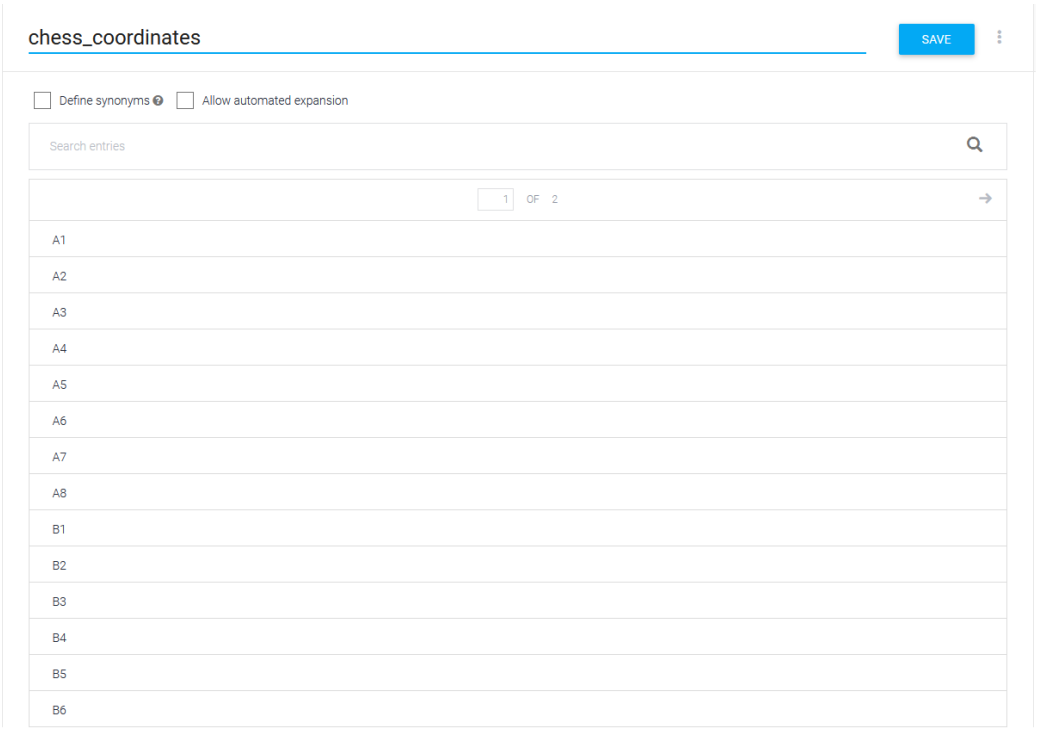

Figure 3. A screenshot of DialogFlow after the definition of the entities of the chess game. The entities are represented by coordinates that the user says to specify a move from a start position to an end position.

5. Use Case Evaluation

In this section, we will perform the architecture and the user experience assessment of the proposed use case.

5.1. Use Case Architecture Evaluation

We initially developed our use case on a MacOS with 8GB RAM and 2,2 GHz Dual-Core Intel Core i7 using Choregraphe 2.1.4 and connected to a hot-spot. Within the same hot-spot, the NAO robot was also connected. Two Amazon EC2 instances were employed to host the SSSCs components. Python was the adopted programming language (for HRICs). Javascript, Python, and Java were adopted for the presented SSSCs. Google Assistant SDK was the voice assistant we employed.

The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them [50].

Three different methods for software architecture evaluations have been developed at the Software Engineering Institute: ATAM (Architecture Tradeoff Analysis Method) [51], SAAM (Software Architecture Analysis Method) [52], and ARID (Active Reviews for Intermediate Designs) [53]. All the others derive from them [54].

Because our use case is at an early development stage we chose SAAM for its evaluation. SAAM appeared in 1994 intending to analyze system qualities early in the life cycle of the software architecture. This allows the developers different architectural options. Some pros that SAAM provides are: (i) potential problems can be identified in the early phase of the development; (ii) the documentation can be improved; (iii) the understanding related to software architecture problems can be enhanced; (iv) it relates different involved stakeholders (architect, maintainer, developer, end-user); (v) through the identification of scenarios it provides feedback for its improvement.

SAAM focuses on modifiability in its various forms (such as portability, subsetability, and variability) and functionality.

- Modifiability allows changes to be made to a system quickly and cost-effectively.
- Portability is the ability of the system to run under different computing environments.
- Subsetability is the ability to support the production of a subset of the system or incremental development.
- Variability represents how well the architecture can be expanded or modified to produce new architectures that differ in specific ways.
- Functionality is the ability of the system to do the work for which it was intended.

The steps of SAAM that we performed within our analysis are the following:

- Identify stakeholders;
- Develop, prioritize and classify scenarios;
- Perform scenario evaluation;
- Reveal scenario interactions;
- Generate overall evaluation.

5.1.1. Identify Stakeholders

Four different types of stakeholders revolve around the proposed use case: end-users, developers, testers, and maintainers. End-users play with the robotic platform we provided. Developers are more interested in the clarity, completeness of the use case architecture and clear interaction mechanisms and will develop on top of it. Testers check error-handling consistency, cohesion, and conceptual integrity. Maintainers keep the platform healthy and locate places of change. Two end-users, two developers, one tester, and one maintainer were selected among master students (end users), senior software engineers working within the university for a spin-off (developers, testers, and maintainers).

5.1.2. Identifying and Classifying Scenarios

Scenarios representing future directions that the system must support were identified. They can be either direct or indirect. Whereas direct scenarios can be executed by the system without any modification, indirect scenarios require edits. In the following, we list each of them.

1. A maintainer with the task of installing the entire use case using a different OS (Windows) than the one (MacOS) used for initial development (DIRECT);
2. A user interacting with the robot using each of the presented modules (DIRECT);
3. A tester that performs all the needed debugging of the use case installed software modules (DIRECT);
4. A developer willing to integrate into the current use case a new module consisting of an HRIC and a SSSC (INDIRECT);
5. A developer willing to integrate into the current use case a new AC (INDIRECT).

5.1.3. Scenarios Evaluation and Interactions

The maintainer was involved in Scenario 1. She used a PC with Windows, an OS different than that used for development. She went through the documentation and was able to install all the required software. After making sure the installation was successful and everything worked properly, she handed over everything to one end-user for deeper testing (as indicated in Scenario 2).

Two end-users were employed for scenario 2. They tested each of the presented modules following the related user documentation. They interacted with the robot using the employed Google Assistant technologies. They both separately spent 2 h each playing with the system. One used the use case with the current settings (i.e., MacOS as OS). The other one used the use case configured by the maintainer as indicated in Scenario 1. They raised a few concerns: on the one hand, each module was correctly triggered by the user. On the other hand, some of the modules returned an unexpected output showing some accuracy problems with the internal function of the module but not with the use case itself. In particular, most of the problems were with the Conversational Agent module that often returned statements too far from the user input statement and the speech-to-text performed by the virtual assistant tool. Besides the accuracy problems of the module (which can be improved with updates on just the module), the two users were satisfied with the overall interaction and usage of the system.

A tester was employed for Scenario 3. He set the debug mode through a field in the configuration file to generate the output of each module. He activated each of the presented modules and looked at their output (in form of logs). The tester did not find any software issue, bugs, or weakness in the system and the communication of the involved components.

Scenarios 4 and 5 were carried out by the two developers. One developed the HRIC and SSSC of an Odd or Even game, whereas the other developed the same game as an AC. They went through the developer guide and the source code identifying the needed components to interact with, their connections, and which interfaces to add. Hence they developed their games. They both had to edit the VAC and the HRI Interface that allowed them to activate the first game by saying the word *Odd or Even* and the other version with the word *Odd or Even AC Game*. When one version of this game was activated, the following interactions of the user affected the modules related to the underlying game version only. As soon as the game was over, the control went back to the main voice assistant interface. The two developers were able to perform those changes and easily install the two modules. Their feedback was related to some improvement (more clear statements related to some technical aspects) in the documentation that would improve the comprehension. The steps followed by the two developers to implement two new modules were added to the developer guide as Hello World examples.

5.1.4. Overall evaluation

Both the two last scenarios affected the VAC and the HRI Interface. On the one hand, in the developed use case the VAC corresponds to the Google Assistant SDK that effectively handles multiple requests. On the other hand, both modules had to access and edit the HRI Interface to include the hooks for the two games. This turned out to be one weakness of the use case given the low modularity of this component. From the developer's point of view, it would have been better to use a mechanism of registration where each developed module

can be exploited independently. This was invaluable feedback and, as such, we are already working to provide such a mechanism in the next version of our use case. Furthermore, the interaction among different stakeholders was definitely a benefit for the resulting version of the documentation and architecture as it helped us to see different sides of the same concept at a different level. Low coupling and high cohesion were two features observed by the developers involved in the last two scenarios.

5.2. User Experience

The functionalities of the robot were demoed for 30 min to 10 participants chosen among students and researchers on different topics of the University of Cagliari (6 males and 4 females). The average of their age was 31.2 with a standard deviation of 3.37. We asked them to perform the following tasks with the robot and for the following amount of time: sentiment analysis (3 min), conversation with the robot (5 min), ask the robot to perform some actions (4 min), ask the robot to identify some objects (3 min), play Bingo (5 min) and chess (5 min). Each user filled a survey: the first part included seven open questions (the first five also included five levels of reply, *very bad*, *bad*, *normal*, *good*, *very good*), whereas the second part was a standard System Usability Scale (SUS) (https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (accessed on 30 June 2021)) questionnaire aimed to evaluate the application's usability. In the following, we summarize their answers to the open questions.

Q1. How do you find the interaction with the robot? Six users gave a *good* score and four *very good*.
Q2. How effectively did the robot classify positive and negative sentences? Five users gave a score of (*very good*), whereas five more gave *normal*, as they found some errors in the ability to recognize the correct sentiment.
Q3. How effectively did the robot perform the action commands you gave? Seven users gave (*good*) mentioning that the robot was able to perform the action from a list. Three users gave *normal* complaining about the robot that was it not always able to recognize their commands. It was easy to fix this problem because users employed peculiar words not covered within the dictionary. The solution was to update the dictionary with the new words.
Q4. How effectively did the robot identify objects? Eight users gave *good* as the robot correctly identified the object they showed. Two users gave *normal* because they used the application at an incorrect distance from the robot. We fixed this by letting the robot say, when the object detection module was triggered, to maintain a certain distance from it.
Q5. How effective was the robot in playing Bingo or chess? All users gave *good* as the robot played both the games smoothly and the users had fun playing with it.
Q6. What are the main weaknesses of the resulting interaction? The only complaint was pointed out by two users mentioning that tiredness of the user might appear when used extensively.
Q7. Can you think of any additional features to be included in the robot interaction? The suggested features were: (1) to have the robot say instructions better when it was turned on and each time a certain module was triggered; (2) the ability to save all the data pertaining to the interaction with the user.

The SUS questionnaire confirmed the good opinion of the users, scoring 83.2/100, equivalent to an A grade, and placed our use case in the 93% percentile rank (percentiles of SUS https://measuringu.com/interpret-sus-score/ (accessed on 30 June 2021)). All users felt very confident in playing with the proposed use case (with an average score of 4.1 ± 0.4) and thought that it was easy to use (4.3 ± 0.3). In addition, they were happy to use it frequently (4.2 ± 0.2) and did not think that it was complex (1.2 ± 0.3) or that they would need the help of a technical person to use it in the future (1.5 ± 0.7).

Considering the first part of the survey, questions Q2, Q3, Q4, Q5 are not specific for the adopted robotic platform and the related responses are general for any robotic platform using the proposed abstraction layer. The underlying applications rely on voice

assistant technologies only and, therefore, are not affected by the employed robotic platform. A robotic wheelchair using voice assistant technology would obtain the same values for Q2, Q3, Q4, Q5. Q1 (and partially Q6 and Q7, corresponding, respectively, to the potential weaknesses of the resulting system and the capabilities that can be added for a better interaction) are two questions which are affected by the used robotic platform and may vary if a robotic wheelchair is used.

6. Conclusions and Future Works

In this paper, we introduced an abstraction layer for HRI that includes voice assistant technologies and that can be plugged into any robotic platform. Among its advantages, it allows reduction in the workload on the robot by using external tools and cloud platforms for running resource-intensive tasks. It also allows the development of applications of different kinds that can augment the interaction between the robot and the user. Moreover, the employment of voice assistant technologies is leveraged for the management of some applications built on top and for the conversation between the robotic platform and the user in the open domain. The scheme provided by the abstraction layer can be instantiated by choosing the robotic platform and technology for each component. One example of such instances is represented by the use case we proposed which uses the NAO robotic platform, the Choregraphe suite, and Google Assistant as voice assistant technology. Moreover, we employed the Google Assistant SDK and the associated developing tools. One example application leveraging Google Assistant and its SDK was developed and included in our proposed use case. Similarly, in our proposed use case, we employed Google's voice services. Furthermore, we evaluated the proposed use case testing possible scenarios with several stakeholders who identified potential weaknesses that we are taking into account for the next release of our architecture. For simplicity, we also evaluated the user experience of the proposed use case through questionnaires we gave to 10 different users. The entire source code for the use case we developed can be freely downloaded (https://github.com/conema/ZAGA (accessed on 30 June 2021)). The presented abstraction layer is general; therefore, if the intention is to use services with more privacy-aware policies, different use cases based on our proposed abstraction layer can be instantiated by adopting solutions such as Snips.ai. A video related to the proposed use case that summarizes the different applications we built on top of NAO is freely available here (https://www.youtube.com/watch?v=tCEIrj6dg_w&feature=youtu.be (accessed on 30 June 2021)). The reader should note that the proposed use case can be applied to any robotic platform including robotic wheelchairs. For the Object Detection Application, the robotic wheelchair must be provided with a camera, whereas for the Mr. Chess Application it should have a screen. Finally, the Robot Action Command should be reset according to the physical movements and positions the robot can assume.

There are several directions that we would like to head in parallel. One of them, the most challenging, and at architecture-level, is related to the development of a smart engine so that each application built on top can be triggered automatically depending on certain events to enrich the HRI between the user with the voice assistant technology by employing certain behavior (e.g., sentiment analysis, object detection) and have the robotic platform performing related actions depending on its sensors, body parts, etc. The idea is to turn any robotic platform closer towards the behavior of a human. One more direction at the use case level is related to the development of a further use case employing ROS (https://www.ros.org/ (accessed on 30 June 2021)), already widespread among a certain number of robotic platforms.

Author Contributions: Conceptualization, D.R.R.; methodology, E.C. and R.A.; software, E.C.; validation, R.A. and E.C.; writing, D.R.R.; supervision, D.R.R.; funding acquisition, D.R.R. and R.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the H2020 project STAR—Novel AI technology for dynamic and unpredictable manufacturing environments (grant number 956573).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research was partially funded by the H2020 project STAR—Novel AI technology for dynamic and unpredictable manufacturing environments (grant number 956573).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feil-Seifer, D.; Mataric, M. Robot-assisted therapy for children with autism spectrum disorders. In Proceedings of the 7th International Conference on Interaction Design and Children, London, UK, 17–24 June 2008; pp. 49–52. [CrossRef]
2. D'Onofrio, G.; Sancarlo, D.; Raciti, M.; Recupero, D.R.; Mangiacotti, A.; Russo, A.; Ricciardi, F.; Vitanza, A.; Cantucci, F.; Presutti, V.; et al. MARIO Project: Experimentation in the Hospital Setting. In Proceedings of the Ambient Assisted Living—Italian Forum 2017, Eighth Italian on Ambient Assisted Living Forum, ForItAAL 2017, Genoa, Italy, 14–15 June 2017; pp. 289–303. [CrossRef]
3. Organisation for Economic Co-operation and Development. *Health at a Glance: Europe 2018*; OECD Ilibrary: Paris, France, 2018; p. 212. [CrossRef]
4. Commision, E. Investing in Health. 2013. Available online: https://ec.europa.eu/health/sites/default/files/policies/docs/swd_investing_in_health.pdf (accessed on 30 June 2021).
5. Singh, A.; Misra, N. Loneliness, depression and sociability in old age. *Ind. Psychiatry J.* **2009**, *18*, 51–55. [CrossRef] [PubMed]
6. More Hospitals Investing in Robots to Cut Costs in the Long Run. 2015. Available online: https://www.healthcarefinancenews.com/news/more-hospitals-investing-robots-cut-costs-long-run (accessed on 30 June 2021).
7. Belpaeme, T.; Kennedy, J.; Ramachandran, A.; Scassellati, B.; Tanaka, F. Social robots for education: A review. *Sci. Robot.* **2018**, *3*. [CrossRef] [PubMed]
8. Panayiotou, M.; Eteokleous, N. *Robotics as Means to Increase Students' Stem Attitudes*; Chapter Projects and Trends; InScience Press: Lisboa, Portugal, 2017; pp. 216–220.
9. Elias. Available online: http://www.eliasrobot.com/ (accessed on 30 June 2021).
10. mBot, Global-Known Robot that Helps Kids Get Smarter as They Play. Available online: https://www.makeblock.com/steam-kits/mbot/ (accessed on 30 June 2021).
11. Recupero, D.R.; Alam, M.; Buscaldi, D.; Grezka, A.; Tavazoee, F. Frame-Based Detection of Figurative Language in Tweets [Application Notes]. *IEEE Comput. Intell. Mag.* **2019**, *14*, 77–88. [CrossRef]
12. Discover What Google Assistant Is. Available online: https://assistant.google.com/ (accessed on 30 June 2021).
13. Siri Does More Than Ever. Even before You Ask. Available online: https://www.apple.com/siri/ (accessed on 30 June 2021).
14. Microsoft Cortana. Available online: https://www.microsoft.com/en-in/windows/cortana (accessed on 30 June 2021).
15. Amazon Alexa. Available online: https://developer.amazon.com/alexa (accessed on 30 June 2021).
16. Build Natural Language Experiences with Wit. Available online: https://wit.ai (accessed on 30 June 2021).
17. Snips. Available online: https://snips.ai/ (accessed on 30 June 2021).
18. Erol, B.A.; Wallace, C.; Benavidez, P.; Jamshidi, M. Voice Activation and Control to Improve Human Robot Interactions with IoT Perspectives. In Proceedings of the 2018 World Automation Congress (WAC), Stevenson, WA, USA, 3–6 June 2018; pp. 1–5. [CrossRef]
19. Faria, B.M.; Reis, L.P.; Lau, N. A Survey on Intelligent Wheelchair Prototypes and Simulators. In *New Perspectives in Information Systems and Technologies*; Rocha, Á., Correia, A.M., Tan, F.B., Stroetmann, K.A., Eds.; Springer International Publishing: Cham, Switzerland, 2014; Volume 1, pp. 545–557.
20. Dialogflow. Available online: https://dialogflow.com/ (accessed on 30 June 2021).
21. Miller, D.P.; Slack, M.G. Design and testing of a low-cost robotic wheelchair prototype. *Auton. Robot.* **1995**, *2*, 77–88. [CrossRef]
22. Marchal-Crespo, L.; Furumasu, J.; Reinkensmeyer, D.J. A robotic wheelchair trainer: Design overview and a feasibility study. *J. Neuroeng. Rehabil.* **2010**, *7*, 40. [CrossRef] [PubMed]
23. Hsu, P.E.; Hsu, Y.L.; Chang, K.W.; Geiser, C. Mobility Assistance Design of the Intelligent Robotic Wheelchair. *Int. J. Adv. Robot. Syst.* **2012**, *9*, 244. [CrossRef]
24. Prassler, E.; Scholz, J.; Fiorini, P. A robotics wheelchair for crowded public environment. *IEEE Robot. Autom. Mag.* **2001**, *8*, 38–45. [CrossRef]
25. Sgouros, N.M. Qualitative Navigation for Autonomous Wheelchair Robots in Indoor Environments. *Auton. Robot.* **2002**, *12*, 257–266. [CrossRef]
26. Hartman, A.; Gillberg, R.; Lin, C.T.; Nandikolla, V.K. Design and development of an autonomous robotic wheelchair for medical mobility. In Proceedings of the 2018 International Symposium on Medical Robotics (ISMR), Atlanta, GA, USA, 1–3 March 2018; pp. 1–6. [CrossRef]
27. Leaman, J.; La, H.M. A Comprehensive Review of Smart Wheelchairs: Past, Present, and Future. *IEEE Trans. Hum. Mach. Syst.* **2017**, *47*, 486–499. [CrossRef]

28. Sukerkar, K.; Suratwala, D.; Saravade, A.; Patil, J.; D'britto, R. Smart Wheelchair: A Literature Review. *Int. J. Inform. Commun. Technol. (IJ-ICT)* **2018**, *7*, 63. [CrossRef]
29. Knote, R.; Janson, A.; Eigenbrod, L.; Söllner, M. The What and How of Smart Personal Assistants: Principles and Application Domains for IS Research. In Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI), Lüneburg, Germany, 6–9 March 2018.
30. Fernando, N.; Tan, F.T.C.; Vasa, R.; Mouzaki, K.; Aitken, I. Examining digital assisted living: Towards a case study of smart homes for the elderly. In Proceedings of the 24th European Conference on Information Systems ECIS, Istanbul, Turkey, 12–15 June 2016.
31. Portet, F.; Vacher, M.; Golanski, C.; Roux, C.; Meillon, B. Design and Evaluation of a Smart Home Voice Interface for the Elderly: Acceptability and Objection Aspects. *Pers. Ubiquitous Comput.* **2013**, *17*, 127–144. [CrossRef]
32. Teixeira, A.; Hämäläinen, A.; Avelar, J.; Almeida, N.; Németh, G.; Fegyó, T.; Zainkó, C.; Csapó, T.; Tóth, B.; Oliveira, A.; et al. Speech-centric Multimodal Interaction for Easy-to-access Online Services—A Personal Life Assistant for the Elderly. *Procedia Comput. Sci.* **2014**, *27*, 389–397. [CrossRef]
33. Gárate, A.; Herrasti, N.; López, A. GENIO: An Ambient Intelligence Application in Home Automation and Entertainment Environment. In Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence: Innovative Context-aware Services: Usages and Technologies, Grenoble, France, 12–14 October 2005; ACM: New York, NY, USA, 2005; pp. 241–245. [CrossRef]
34. Gordon, M.; Breazeal, C. Designing a Virtual Assistant for in-Car Child Entertainment. In Proceedings of the 14th International Conference on Interaction Design and Children, Boston, MA, USA, 21–24 June 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 359–362. [CrossRef]
35. Hauswald, J.; Laurenzano, M.A.; Zhang, Y.; Li, C.; Rovinski, A.; Khurana, A.; Dreslinski, R.G.; Mudge, T.; Petrucci, V.; Tang, L.; et al. Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers. In Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems, Istanbul, Turkey, 14–18 March 2015; ACM: New York, NY, USA, 2015; pp. 223–238. [CrossRef]
36. Sarikaya, R.; Crook, P.A.; Marin, A.; Jeong, M.; Robichaud, J.P.; Celikyilmaz, A.; Kim, Y.B.; Rochette, A.; Khan, O.Z.; Liu, X.; et al. An overview of end-to-end language understanding and dialog management for personal digital assistants. In Proceedings of the 2016 IEEE Spoken Language Technology Workshop (SLT), San Diego, CA, USA, 13–16 December 2016; pp. 391–397. [CrossRef]
37. Usachev, D.; Khusnutdinov, A.; Mazzara, M.; Khan, A.; Panchenko, I. Open source platform Digital Personal Assistant. *arXiv* **2018**, arXiv:1801.03650.
38. MQTT: The Standard for IoT Messaging. Available online: https://mqtt.org/ (accessed on 30 June 2021).
39. Atzeni, M.; Recupero, D.R. Deep Learning and Sentiment Analysis for Human-Robot Interaction. In Proceedings of the Semantic Web: ESWC 2018 Satellite Events—ESWC 2018 Satellite Events, Heraklion, Crete, Greece, 3–7 June 2018; Revised Selected Papers, pp. 14–18. [CrossRef]
40. Dridi, A.; Reforgiato Recupero, D. Leveraging semantics for sentiment polarity detection in social media. *Int. J. Mach. Learn. Cybern.* **2017**, *10*, 2045–2055. [CrossRef]
41. Atzeni, M.; Dridi, A.; Recupero, D.R. Using frame-based resources for sentiment analysis within the financial domain. *Prog. AI* **2018**, *7*, 273–294. [CrossRef]
42. Atzeni, M.; Recupero, D.R. Fine-Tuning of Word Embeddings for Semantic Sentiment Analysis. In Proceedings of the Semantic Web Challenges—5th SemWebEval Challenge at ESWC 2018, Heraklion, Greece, 3–7 June 2018; Revised Selected Papers, pp. 140–150. [CrossRef]
43. Atzeni, M.; Dridi, A.; Recupero, D.R. Fine-Grained Sentiment Analysis on Financial Microblogs and News Headlines. In Proceedings of the Semantic Web Challenges—4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, 28 May–1 June 2017; Revised Selected Papers, pp. 124–128. [CrossRef]
44. Gangemi, A.; Presutti, V.; Recupero, D.R. Frame-Based Detection of Opinion Holders and Topics: A Model and a Tool. *IEEE Comp. Int. Mag.* **2014**, *9*, 20–30. [CrossRef]
45. Recupero, D.R.; Presutti, V.; Consoli, S.; Gangemi, A.; Nuzzolese, A.G. Sentilo: Frame-Based Sentiment Analysis. *Cogn. Comput.* **2015**, *7*, 211–225. [CrossRef]
46. Dridi, A.; Atzeni, M.; Recupero, D.R. Bearish-Bullish Sentiment Analysis on Financial Microblogs. In Proceedings of the 3rd International Workshop at ESWC on Emotions, Modality, Sentiment Analysis and the Semantic Web Co-Located with 14th ESWC 2017, Portroz, Slovenia, 28 May 2017.
47. Atzeni, M.; Reforgiato, Â.; Recupero, D. Multi-domain sentiment analysis with mimicked and polarized word embeddings for humanoid robot interaction. *Future Gener. Comput. Syst.* **2020**, *110*, 984–999. [CrossRef]
48. Reforgiato Recupero, D.; Spiga, F. Knowledge acquisition from parsing natural language expressions for humanoid robot action commands. *Inf. Process. Manag.* **2020**, *57*, 102094. [CrossRef]
49. Reforgiato Recupero, D. Technology Enhanced Learning Using Humanoid Robots. *Future Internet* **2021**, *13*, 32. [CrossRef]
50. Bass, L.; Clements, P.; Kazman, R. Software Architecture in Practice. 2003. Available online: https://books.google.ro/books?hl=en&lr=&id=ZY6UZTjBnGQC&oi=fnd&pg=PA1&dq=Bass,+L.%3B+Clements,+P.%3B+Kazman,+R.+%5Cemph%7BSoftware+Architecture+in+Practice%7D%3B++%5Chl%7B2003.%7D&ots=gsptTsR_rh&sig=YhRFJCTVL4ldBqRggttYO-SIBck&redir_esc=y#v=onepage&q&f=false (accessed on 30 June 2021).

51. Kazman, R.; Klein, M.; Barbacci, M.; Longstaff, T.; Lipson, H.; Carrière, S. The Architecture Tradeoff Analysis Method. In Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193), Monterey, CA, USA, 6 August 2002; pp. 68–78. [CrossRef]
52. Kazman, R.; Bass, L.; Abowd, G.; Webb, M. SAAM: A method for analyzing the properties of software architectures. In Proceedings of the 16th International Conference on Software Engineering, Sorrento, Italy, 16–21 May 1994; pp. 81–90.
53. Clements, P. *Active Reviews for Intermediate Designs*; Software Engineering Institute, Carnegie Mellon University: Pittsburgh, PA, USA, 2000; p. 26.
54. Ali Babar, M.; Gorton, I. Comparison of Scenario-Based Software Architecture Evaluation Methods. In Proceedings of the 11th Asia-Pacific Software Engineering Conference, Busan, Korea, 30 November–3 December 2004; pp. 600–607. [CrossRef]