Cognitive Systems Research

Towards an integrated robotics architecture for social inclusion – The RAPP paradigm

Emmanouil G. Tsardoulias[a], Athanassios M. Kintsakis[b*], Konstantinos Panayiotou[d], Aristeidis G. Thallas[a], Sofia E. Reppou[d], George G. Karagiannis[d], Miren Iturburu[e], Stratos Arampatzis[f], Cezary Zielinski[c], Vincent Prunet[g], Fotis E. Psomopoulos[a], Andreas L. Symeonidis[a] and Pericles A. Mitkas[b]

[a]*ITI – Information Technologies Institute, CERTH – Centre for Research and Technology Hellas, Thessaloniki 57001, Greece*
[b]*Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece*
[c]*Institute of Control and Computation Engineering, Warsaw University of Techonology, PL-00-665 Warszawa, Poland*
[d]*Ormylia Foundation, Ormylia 63071, Greece*
[e]*Matia Instituto Gerontologico, Donostia-San Sebastian, Spain*
[f]*Ortelio Ltd, Coventry, CV1 2TT, UK*
[g]*Inria Sophia Antipolis - Méditerranée, France*

Abstract

Scientific breakthroughs have led to an increase in life expectancy, to the point where senior citizens comprise an ever increasing percentage of the general population. In this direction, the EU funded RAPP project "Robotic Applications for Delivering Smart User Empowering Applications" introduces socially interactive robots that will not only physically assist, but also serve as a companion to senior citizens. The proposed RAPP framework has been designed aiming towards a cloud-based integrated approach that enables robotic devices to seamlessly deploy robotic applications, relieving the actual robots from computational burdens. The Robotic Applications (RApps) developed according to the RAPP paradigm will empower consumer social robots, allowing them to adapt to versatile situations and materialize complex behaviors and scenarios. The RAPP pilot cases involve the development of RApps for the NAO humanoid robot and the ANG-MED rollator targeting senior citizens that a) are technology illiterate, b) have been diagnosed with mild cognitive impairment or c) are in the process of hip fracture rehabilitation. Initial results establish the robustness of RAPP in addressing the

[*] Corresponding author. Tel.: +30-2310-996349; fax: +30-2310-996398.
E-mail address:akintsakis@issel.ee.auth.gr

needs of end users and developers, as well as its contribution in significantly increasing the quality of life of senior citizens.

Robotic applications; Cloud Robotics; Robotic Architectures; Assistance Robotics; Social Robotics

1. Introduction

It is becoming apparent that technological and medical advances have brought a major demographic change in our society by increasing the overall life expectancy and consequently the percentage of senior citizens to the general population (Lutz, Sanderson, & Scherbov, 2008). As a result, current social infrastructures and services are struggling to keep up with the new standard. In the near future, older people requiring support in their daily life are expected to largely outgrow the supply of potential caregivers. Socially interactive robots can effectively address the issue not only by physically assisting people but also by functioning as a companion (Bemelmans, Gelderblom, Jonker, & De Witte, 2012; Hutson, Lim, Bentley, Bianchi-Berthouze, & Bowling, 2011).

The ever increasing robot sales figures pose a clear indication of a new rising trend where the number of robots participating in everyday life, as well as their use cases and roles undertaken, are rapidly expanding. In anticipation of social robots invading everyday life, the *RAPP project* introduces robots as software platforms in an attempt to attract developer interest and lower the costs involved, allowing for increased autonomy, social inclusion and overall quality of life of senior citizens (Mitkas, 2015). The RAPP project, aptly named "Robotic Applications for Delivering Smart User Empowering Applications", aims to provide an open-source software platform supporting the creation and delivery of robotic applications (RApps).

RAPP provides a robot agnostic approach by introducing a middleware stack with added functionalities suitable for different kinds of robots. In this way, developers will be able to utilize a common API (RAPP Robot API) in order to access the robot's sensors and actuators. A number of high level services and functionalities for implementing RApps are also offered by the RAPP Platform and are accessible through the RAPP API. The RAPP Platform offers a cloud-based solution that lifts the underlying robotic hardware computational and storage limitations and enables advanced machine learning operations, distributed data collection and processing, as well as knowledge sharing among robots. The developed RApps are hosted in the proposed RAPP Store, which is ultimately expected to have an important effect in the robotic application market. The adoption of a common API will provide developers with the versatility needed to address people with different needs, capabilities and expectations, while simultaneously respecting their privacy and autonomy.

The RAPP project also aims to develop robotic applications that encourage the social, emotional and cognitive empowerment of people at risk of exclusion and in particular older people with accompanying physical, social or cognitive disabilities. The solutions offered enable older people to remain socially active and increase their independence and autonomy, while at the same time relieving caregivers from tedious tasks and providing the necessary tools to the medical community for a better assessment of the functional and cognitive status of the patient.

The rest of the paper is organized as follows. Chapter 2 includes an analysis of related work and the state of the art. Chapter 3 presents the RAPP architecture in detail and chapter 4 illustrates the first results of the

RAPP pilot cases, as well as a developer evaluation of the RAPP software framework. A brief discussion section follows in chapter 5 along with future work and finally chapter 6 concludes the paper.

2. State of the art

As the implementation of the RAPP project is a combination of robotic systems, architectures and algorithms, cloud robotics and application distribution and deployment methods, a brief overview of the main representatives of each category is provided. Regarding robotic systems, one of the most famous representatives is the Robotic Operating System (ROS), being both a framework and a middleware oriented towards developing robot software in a modular way. It is a collection of tools, libraries and nomenclatures, providing the necessary abstraction for a robotic developer to be able to effortlessly create complex multi-process robotic applications. It is described as a meta-operating system (Quigley et al., 2009), as it provides standard system operating services such as hardware abstraction, low-level device control, implementation of commonly used functionalities and message passing between processes, as well as package management. It is fully distributed and asynchronous, as it allows for transparent node (process) execution on heterogeneous robotic systems or computers. ROS is currently the state-of-the-art in robotic middleware, as evidenced by the great number of publications and the amount of diverse research it is involved in. For example, in (Joyeux & Albiez, 2011), ROS is investigated as the tool to traverse from robotic components to whole systems. Additionally, the distributed characteristic of ROS, as well as some of its ports to the JavaScript language (Osentoski et al., 2011), allows it to be the link between intelligent environments and the “Internet of Things” (Roalter, Kranz, & Möller, 2010). Elkady, Joy and Sobh (Elkady, Joy, & Sobh, 2011), use ROS as a plug-and-play middleware for sensory modules, actuator platforms and task descriptions in robotic manipulation platforms, whereas in (Beetz, Mösenlechner, & Tenorth, 2010), Beetz et. al. use it as the basis for the creation of another framework (CRAM), a Cognitive Robot Abstract Machine for everyday manipulation in human environments. One of the most important aspects of ROS is that due to its modularity and standardization via offering common data structures, the open-source robotics community constantly provides software packages that can be directly installed and employed in a great variety of diverse robotic devices. This enables ROS and its ecosystem to pose as a low-level robotic application store, one where all apps are free and distributed through standard Linux package managers, such as apt-get (Gerkey & Garage, 2009; Nielsen, Bøgild, Jensen, & Bertelsen, 2011).

There already exist a few robotic application stores, as robots slowly emerge as household devices. One of the most famous stores is the RobotAppStore (Wang, Johnston, & Williams, 2012), officially launched in 2011. It enables robotic developers to upload software for any robot they prefer, regardless of the programming language used, as well as define the pricing of their product. Other developers or end-users can then purchase, download and install the software on their robots. Another robot app-store paradigm is Aldebaran’s NAO Store[†], which solely hosts applications installable on the NAO robot. The main difference between NAO Store and RobotAppStore, apart from the targeted robotic devices, is that NAO Store is created in a way that supports one-click installation of the respective application in NAO, whereas the installation of applications downloaded from RobotAppStore is manual.

Nonetheless, the “robotic revolution” cannot take place based solely on efficient software distribution and deployment. A major factor in the incorporation of robotic devices in our everyday lives, is the so called

[†] https://store.aldebaran-robotics.com/

“Cloud Robotics”, essentially describing a world wide web for robots, a place where robots can collaborate by exchanging knowledge about their environments, contributing to increasing the collective robotic cognition (Wang et al., 2012). Several projects exist aiming to establish a common knowledge pool for the entirety of robots, one of the most important being RoboEarth, an FP7-ICT project (Waibel et al., 2011). As RoboEarth describes, “it is a World Wide Web for robots; a giant network and database repository, where robots can share information and learn from each other regarding their behavior and environment” (Tenorth, Klank, Pangercic, & Beetz, 2011). Additionally, it employs KnowRob (Tenorth & Beetz, 2013), an ontology-based knowledge database for robots, combining knowledge representation and reasoning methods with techniques for acquiring knowledge and grounding it in a physical system that can serve as a common semantic framework for integrating information from heterogeneous sources. Furthermore, it supports different deterministic and probabilistic reasoning mechanisms, clustering, classification and segmentation methods and includes query interfaces as well as visualization tools. Finally, the RoboEarth Cloud Engine nicknamed “Rapyuta” (Hunziker, Gajamohan, Waibel, & D'Andrea, 2013) has been presented as a Platform-as-a-Service (PaaS) framework, allowing robots to offload heavy computations by providing secure customizable computing environments in the cloud, while allowing easy access to the RoboEarth knowledge repositories.

Another example of collective robot knowledge acquisition is displayed by the ROBOHOW.COG project, concerning web-enabled and experience-based cognitive robots that learn complex everyday manipulation tasks. Its main goal is to enable robotic devices to competently perform everyday human-scale manipulation activities, both in human working and living environments. It achieves the goal of building a cognitive robot that autonomously performs complex tasks and extends its ensemble of such, by creating new skills using web-enabled and experience-based learning, as well as by observing humans (Tenorth et al., 2011; Tenorth, Nyga, & Beetz, 2010). It must be stated that ROBOHOW uses KnowRob as well. Other similar projects include PEIS-Ecology (Saffiotti et al., 2008), combining ideas from the field of autonomous robotics and ambient intelligence towards creating integrated house social robotic systems, URC (Ubiquitous Robotic Companion) (Ha, Sohn, Cho, & Yoon, 2005), where the main idea is that robots have always access to services regardless of environmental alterations and NRS (Networked Robot Systems) (Sanfeliu, Hagita, & Saffiotti, 2008) which provides collaboration between physical robots, environment sensors / actuators and humans, with the help of autonomous network-based systems.

Model driven software development is a valid approach for coping with the complexity of robotic systems. In the SMARTSOFT (Schlegel, Haßler, Lotz, & Steck, 2009) meta model, robotic software components are modeled in UML, external interfaces are defined via interaction patterns and execution is modeled after state automata. The resulting platform independent models can then be transformed into code templates. The BRICS project (Bruyninckx et al., 2013), defines a component model that aims to provide developers with as much structure as possible in their development process while avoiding indulging into application-specific details. In (Zander et al., 2016), the ReApp architecture presented combines model-driven engineering with semantic technologies in order to facilitate the development and reuse of ROS-based components and applications. The proposed model-driven tooling approach empowers developers to work at higher abstraction levels and fosters automatic code generation.

RAPP differentiates from the aforementioned technologies, as it tries to combine their functionalities into a single integrated system implementation. Specifically, the RAPP ecosystem consists of a *web (cloud) part* containing the RAPP Store (Robotic Applications Store), as well as the knowledge pool and inference methodologies and the *robot-component* that represents each supported robotic device able to download and install RApps. The cloud-part also provides services for employment of heavy duty computations on the web instead of on the robot. The component bridging together the entire system is HOP, a toolset for programming the Web of Things (Serrano & Berry, 2012; Serrano, Gallesio, & Loitsch, 2006; Serrano & Queinnec, 2010). HOP is a multitier programming environment built on top of web protocols and languages. It orchestrates data and command transfer among objects which could be web services and user interface components. It is typically used to coordinate home automation and robotic environments for assisted living. In this case, the environments can be seen as an aggregation of communicating objects (sensors and active components such as robots) which are discovered, identified and linked to distributed client/server HOP software modules. The different modules comprising the RAPP architecture have been defined either on a technology level (i.e. different implementation layer), or at a conceptual level (i.e. data mining module).

3. The RAPP Architecture

RAPP was designed as a distributed architecture consisting of two major layers that are further divided into subcomponents and specific systems. These are the RAPP Platform (residing on the cloud) and the Robot Platform (residing at the software level of each robot supported by RAPP). The subcomponents within each layer are grouped into entities named Agents and include the Platform, Cloud, Dynamic and Core Agents. Robotic applications (RApps) are executed on the RAPP architecture utilizing one or more Agents. In Figure 1 the layers and Agents of the RAPP architecture are presented and are further analyzed in the following subsections.

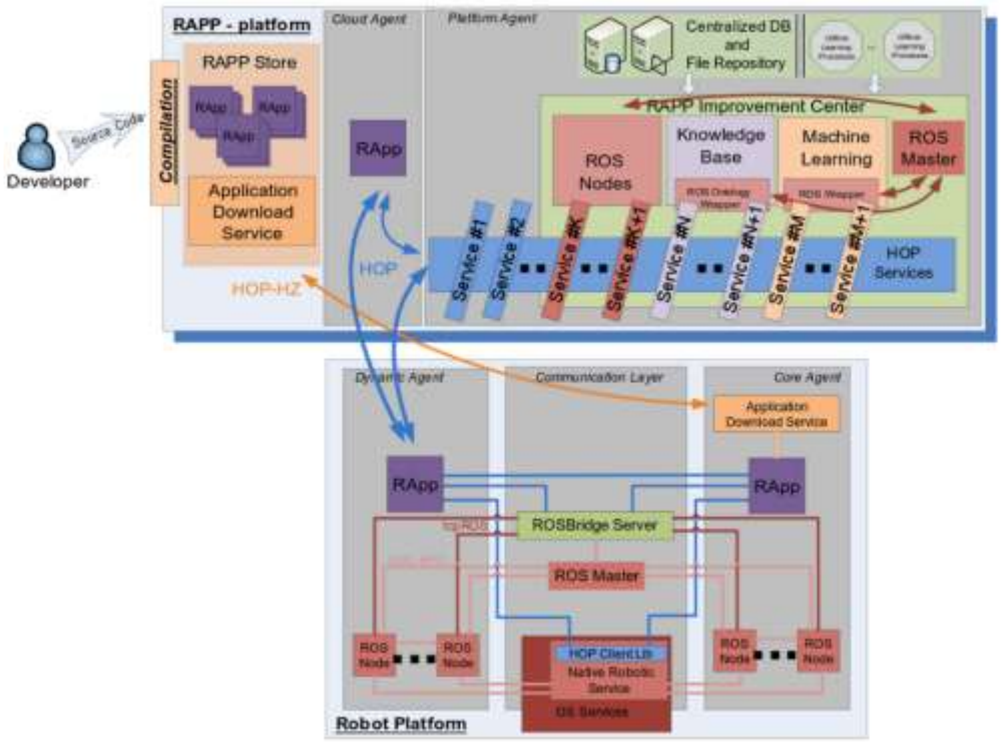

Fig. 1. An overview of the RAPP Platform architecture along with its major layers and respective subcomponents.

3.1. Robotic applications (RApps)

One of the most important entities in the RAPP framework is a Robotic Application or *RApp*. A RApp is simply an instance of an application provided by a developer, and is distributed through the RAPP Store after its submission and the appropriate validation process. An actual application may be distributed in nature; however, it shall always be executed mainly on the robot and only the distributed parts may have components or utilize services that exist on the cloud. Assuming that a RApp exists in a host/guest fashion, the host exists in the robot part and is the main robot controller, whilst the guest may exist in the cloud. The offered distributed execution is a great advantage, since algorithms that cannot be naturally executed in robots due to resources limitations can now be utilized in the cloud (i.e. CUDA employment, MPI processing etc.). RApp implementations may range from single robot simple applications to advanced multi-robot cooperation real-life paradigms.

3.2. RAPP Store

A distinct component within the RAPP Platform is the RAPP Store, which mainly acts as the web-interface to the rest of the Framework. Every new robot owner can register to the RAPP Store and subsequently access the RAPP infrastructure. New robots are also registered to the RAPP Platform and are granted access to the RAPP Platform, the RAPP Cloud Agents and RApps. The latter ones are offered in the form of precompiled binaries.

The RAPP Store provides a RESTful API, allowing a user to authenticate, retrieve information concerning the available RApps, as well as download precompiled RApp binaries to their robot. Finally, the RAPP Store acts as a repository for user submitted RApps. Every user-developer is allowed to create their own RApps which are in turn uploaded to the RAPP Store and are made available to all RAPP-registered robots. The developer is also allowed to specify RApp meta-information such as tags and which robotic platforms the specific RApp supports.

3.3. RAPP Core and Dynamic agents

The Core Agent layer (lower right side of Fig.1), is considered to be pre-installed on each robot and is divided in three parts with distinct functionalities, them being the *Core-Store collaboration module*, the *Robot-aware API* and the *Core functionalities (Szlenk, Zieliński, Figat, & Kornuta, 2015; Zielinski et al., 2015).*

The Core-Store collaboration module uptakes the tasks of acquiring and deploying the robotic applications. Each robot can invoke the RAPP Store RESTful API in order to obtain knowledge of the RApps supported by the specific robot, or even search among these RApps by keywords. This way, the corresponding Core Agent acquires information concerning specific RApps and retrieves their download URLs. Once a RApp is downloaded via the provided in-robot Application Download Service, the application is automatically deployed based on its type. The types supported are the same as the ones in the RAPP Store, i.e. C++, JS, Python or ROS (C++/Python) applications. Following its successful deployment, the RApp can be treated as a *Dynamic Agent*, since it essentially is a controller that can be dynamically deployed or terminated by the Core Agent. It should be stated that when a Dynamic Agent is deployed, the Core Agent foregoes control of the robot until the corresponding RApp is terminated or stopped.

The Robot-aware API is a collection of service calls in all three programming languages (C++, JS, Python), providing access to low level robotic functionalities, those being the robot's sensors and effectors. This API is robot-aware since each robot has heterogeneous sensors with different nominal properties, thus a per-robot-type API is needed. Nevertheless, each robot-aware API is implemented based on an abstract API toolset by the ABC programming model (Abstract Class). This way, each RApp (or Dynamic Agent) can be developed invoking the abstract API, thus making the application robot agnostic.

The Core functionalities comprise robotic algorithms that cannot be remotely or in a distributed fashion executed due to hardware, resource or temporal restrictions. Some examples are motion controllers or object tracking algorithms requiring a close-loop feedback between cameras and object detection algorithms. Of course, since each robot type is different in nature and in resources (computational, memory or kinematic), it must have its own Core Agent providing different Core functionalities and robot-aware API services.

Finally, when no Dynamic Agent is operating, the Core Agent becomes the controller once more. In that case and based on the particular robot, heterogeneous idle mechanisms may be deployed, waiting external (or internal) events to trigger the execution of RApps.

3.4. The RAPP Platform (RIC)

The RAPP Improvement Centre (RIC), alternatively denoted as Platform Agent, is a cloud-based open source platform including a number of supporting services that provide high level functionalities. These services can be invoked from any other part of the Platform or Robot side and efficiently address a number of functionalities requiring a significant level of expertise to implement and also substantial computational resources for their execution. To address this necessity, state of the art solutions were implemented and are offered in user friendly web services that RApp developers can easily utilize to enhance their application by invoking them off-the-shelf. The RAPP Platform services are summarized as follows and their component diagram is depicted in Figure 2.

RAPP Ontology: The RAPP repository and knowledge management system use a common ontology for all robots participating in the RAPP ecosystem. An ontology-based approach was chosen for knowledge representation, as it provides a versatile schema-free approach capable of deriving knowledge either directly or indirectly through the semantic links between objects and classes. Our ontology scheme is based on the KnowRob robotic ontology. The KnowRob ontology was enhanced by creating new classes derived from the OpenAAL ontology (Wolf et al., 2010) as well as others as needed for the creation of the desired RApps. The core code of KnowRob was also modified in order to perform integrity checks and allow for extra functionalities. Higher level Prolog functions were developed that address common ontology queries and use cases. As the knowledge base expands, the collective knowledge of all robots will grow, allowing for advanced collaboration and knowledge sharing. By effortlessly tapping into existing knowledge, new robots can instantly reach the top of their learning curve, thus achieving an exponential learning rate.

RAPP Centralized Database: The Centralized Database is a MySQL database that serves as the central RAPP Platform data repository. Personal user information including names, contact information and ownership of robots along with RApp specific information including application name, version and supported robots are stored within. The stored data will be treated according to the specifications and guidelines outlined in the privacy and security management protocol. Finally, the RAPP Platform MySQL database contains information on the application tokens needed by the RApps to perform calls to the Platform, as well as deployment data for the submitted Cloud Agents.

RAPP Cognitive Exercise: The RAPP Cognitive exercise system provides a means of performing basic cognitive exercises (Kintsakis, Reppou, Karagiannis, & Mitkas, 2015) in an attempt to exercise and improve the user's arithmetic, reasoning and awareness cognitive skills. A number of tests have been implemented for each category and in different difficulty variations. A robot equipped with speakers and a microphone dictates the questions to the user along with their predefined answers and records the user selected response. A user performance history is being kept in the ontology that aids in keeping track of the user's cognitive test performance and in adjusting the difficulty of the tests he is presented with. The test performance history of the user is stored in the ontology and serves as a basis for adjusting the difficulty of the selected tests according to the user's specific needs, accurately reflecting his individual cognitive strengths and weaknesses. Furthermore, tests are selected in a least recently used fashion in order to avoid repetition and preserve the user's interest.

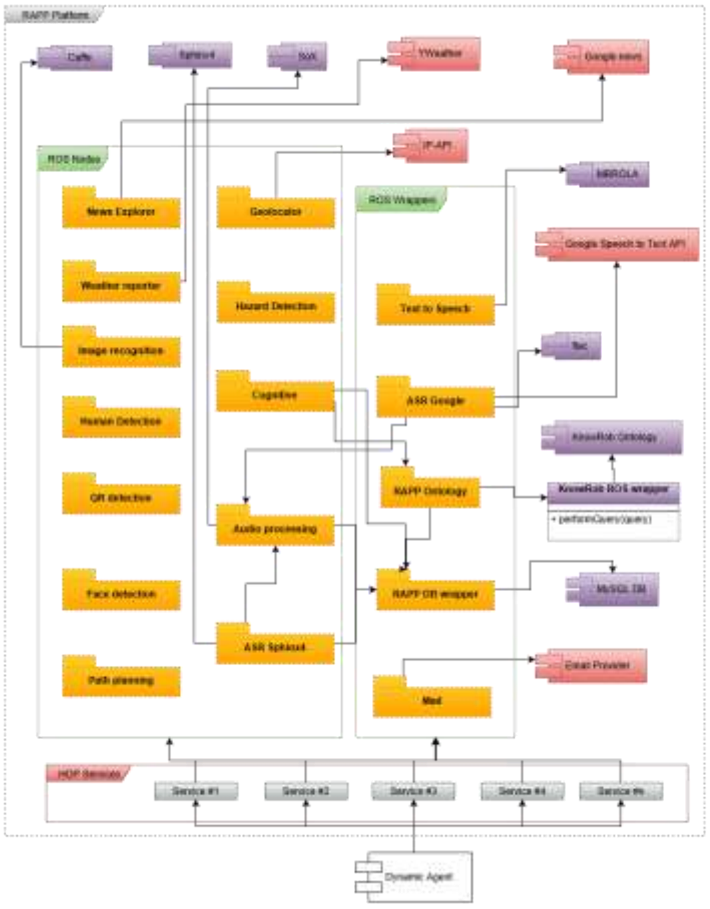

Fig. 2. The component diagram illustrating the RAPP Platform services, their interrelationships and their internal and external component requirements.

RAPP Image Recognition: The RAPP image recognition service utilizes Caffe, an open source framework (Jia et al., 2014) for state-of-the-art deep learning algorithms along with a collection of reference models, developed and maintained by the Berkeley Vision and Learning Center (BVLC). In our use case, the BVLC reference pre-trained model for image recognition was used as it is capable of identifying a substantial number of objects found within the household environment.

The image recognition service largely depends on the ontology knowledge management service in order to extract meaningful semantic knowledge following the identification of an object. Their synergistic use can efficiently tackle scenarios involving robot interaction with objects following the identification of an object.

RAPP Speech to text/Text to Speech: As speech recognition is required for all RApps regardless of the actual robot at hand, we adopted a service based approach by installing the Sphinx-4 (Walker et al., 2004) speech recognition module on the RAPP Platform. Sphinx-4 is a state of the art flexible speech recognition system, capable of multi-language extension and good performance on limited or generalized language models. The speech recognition service can accept uploaded or streamed audio and the ASR can be configured towards language or vocabulary specific decisions.

Sphinx-4 contains a built-in acoustic model, language model and a phonetic dictionary. Furthermore, it can utilize a grammar imposing a set of rules that provide limited but highly accurate word recognition, ideal for command and control use cases. For robots that are equipped with microphones but do not inherently support text to speech, a RAPP Platform service was created utilizing the eSpeak[‡] speech synthesis library as well as the MRBROLA (Dutoit, Pagel, Pierret, Bataille, & Van der Vrecken, 1996) project specializing in altering speech synthesis voices.

A special audio processing service performs the necessary supporting operations to the speech recognition modules. It supports conversion of the audio file to the supported by the RAPP speech detection Sphinx-4 service input format of single channel 16kHz sample rate and 16 bit little endian .wav file. As the robot captured audio is often noisy or distorted, a de-noising procedure must take place before the speech recognition takes place. Since each robot creates recordings with noise patterns (party based on the proximity of the microphone to the robot's cooling fans) of varying spectral characteristics, a personalization procedure must be performed by storing recording samples of each robot and extracting the DFT coefficients of the noise. The de-noising procedure utilizes the SoX Unix audio library along with custom techniques and is offered as a service by the RAPP Audio Processing Platform service.

RAPP Google ASR: The RAPP Google ASR provides a web-based speech to text service based on Google ASR. Two modes are supported, one-shot speech or streaming and the results are returned as a list of hypotheses along with the most dominant one. Unfortunately, due to the limitations imposed, the input stream cannot be longer than 10-15 seconds and no more than 50 requests per day can be submitted. In addition, the service only operates on single channel 16 kHz flac files, mandating that a format conversion takes place. For these reasons, this service was abandoned in favor of the Sphinx-4 Speech recognition service described above; its availability however has been retained in case a developer specifically desires its function.

[‡] http://espeak.sourceforge.net/

RAPP Hazard Detection: The hazard detection module provides two services, detecting if the house lights are on or off and the state of a door (open or closed). The light status detection service takes as input an RGB image and returns an estimation about whether the light is turned on or off, by computing a comparison of the average brightness of eight surrounding regions in comparison to a ninth image region. On the other hand, the door state service is returning a confidence value of door being left open, by analyzing vertical and horizontal segments produced by a line extractor (e.g. RANSAC applied on the result of an edge detector). These algorithms are utilized in the cases of hazard detection in houses where elders live, aiming at supporting their independence (Zielinski et al., 2015).

RAPP Human Detection: The human detection module utilizes features in order to detect humans or human parts in an RGB image. Significant research has been performed investigating the performance of known methods such as Haar-like image features with cascade classifiers, HOG detectors, Daimer classifiers and Latent SVM detectors. Among the tested human detectors in OpenCV, the best one is the Latent SVM detector (called as "Discriminatively Trained Part Based Models for Object Detection") (Felzenszwalb, Girshick, McAllester, & Ramanan, 2010). It provides good results for virtually any position of a human, under different hand positions, even when a large part of the human posture is hidden. Good detection results have been achieved even with structured background and low image contrast. The only difficulties appear if blurred images are processed.

RAPP Path Planning: A simplified version of the motion planning problem is planning a collision free path for a robot in the environment, provided that a map exists. Low-dimensional path planning problems can be efficiently solved with grid-based algorithms that overlay a grid on the top of any geometric map. Our approach utilizes a 2D occupancy grid map consisting of equally-sized grid cells that are either free or occupied. The Occupancy Map can be extracted from the 3D OctoMap (Wurm, Hornung, Bennewitz, Stachniss, & Burgard, 2010) of the environment. There, a path planning algorithm is executed which takes as input the robot and target poses, as well as the OGM (Occupancy Grid Map) and produces a path, consisting of a series of 2D points (Dudek, Szynkiewicz, & Winiarski, 2016).

RAPP Face Detection: The RAPP face detection service is capable of detecting faces in arbitrary images according to the latest trends in image manipulation. This functionality is critical to the ability of the robots to interact with people. Detecting humans in the house for interaction or tracking purposes, recognizing the user for personalization purposes, face tracking for mimicking human motions and emotion recognition by facial characteristics are some the most representative use cases. For real-time applications, the face detection service lowers the resolution of the image in order to maintain a higher frame rate per second.

RAPP Mail Management: The mail management service is tasked with being the proxy between the end user and the mail provider. This service provides an easy and effortless way for older people to actively send and receive emails. The service supports email providers that allow access to their SMTP servers for email transmission and their IMAP servers for email reception. The user's email credentials are stored in the RAPP Core Agent, i.e. in the robot itself.

RAPP Geolocator: The Geolocator service provides area localization and location information utilizing the IP-API[§]. It performs an IP address based search and returns location information including name of city,

[§] http://ip-api.com/

country, region, zip code, latitude and longitude. The service is used mainly by other RAPP Platform services that require location specific information, like the RAPP News Explorer and RAPP Weather Reporter analyzed below.

RAPP News Explorer: The RAPP news explorer service enables RApps to perform news searches on websites/engines. As access to newspapers and magazines can be prohibitive to older people due to vision or motor disabilities, this service can keep them informed, entertained and up to date by streaming news to their robot according to their predefined topics of interest. Depending on the actual robot, content can either be delivered as an audio file which the robot will playback or be read directly from a display in case the robot is equipped with one. The currently supported news source is the Google News API[**].

RAPP Weather Reporter: Older people are particularly interested in the weather as it can be a contributing factor to their health issues and directly affect their activities. This RAPP service is capable of retrieving weather forecasting data for the given location of the robot. Again, the content is either played back by the robot as an audio file or displayed on the robot's screen. Summarized weather reports are also supported as well as personalized alerts to the user depending on his specific health issues or needs. The currently supported weather forecast source is YWeather[††].

RAPP QR Detection**:** The QR code service acts as a wrapper to the popular ZBar[‡‡] library, in conjunction to OpenCV for image manipulation in order to scan QR codes. QR codes are significantly used in the RAPP project in visual tagging of objects, and semantic spatial information like rooms in order to assist in the mapping, localization and navigation processes of the robots.

3.5. RAPP Platform Web services

RAPP Improvement Center (RIC) functionalities can be utilized by robots via standard web services. The service layer has been developed using HOP consisting of a web server implementation, an http/https server, and the web services developed in Hop.js framework. Web services are executed within forked server-side web workers, each of which can include more than one web services allowing concurrent execution. Currently, the RAPP Platform hosts 27 web services, allowing robot platforms to gain access to the RIC functionalities described in 3.4.

RAPP Platform's communication protocols are divided into two parts; the internal communication protocol and the external communication protocol. The first covers all communications between the service layer and the RAPP Improvement Center (RIC), using web sockets, being the *web socket transport layer*. For this purpose, the ROS part of the Platform hosts a web socket server (rosbridge-websocket-server), providing a JSON API to ROS functionalities for non-ROS programs through a web socket transport layer. On the other side, each web service is a client to the web socket server. To simplify and automate the creation of ROS connections from within web service implementations, a lightweight, event-driven, rosbridge client library has been developed in the JavaScript programming language. This serves communication between individual physical robot platforms and the RAPP Platform cloud infrastructure over the network layer. The service

[**] https://developers.google.com/news-search/
[††] https://pypi.python.org/pypi/yweather/
[‡‡] http://zbar.sourceforge.net/

layer, a Platform-as-a-Service (PaaS) model in cloud computing platforms, is often referenced as the communication server of the platform. Both layers allow asynchronous bipolar communications, whilst web service responses are asynchronous http responses. This way we allow robotic platforms to request for cloud services while performing other tasks. Asynchronous platform service invocations are also supported by the respected RAPP-Platform-APIs. A graphical representation of the described process is evident in Figure 3.

Robotic platforms can access RAPP Platform services using HTTP POST requests, as most of the services require an arbitrary amount of data to be sent to the Cloud for processing, like image and audio data files. The HOP Web Server has been configured to accept *application/x-www-urlencoded* and *multipart/form-data* form submissions. Instead of using http post parameters, we use a JSON tree to dump data. The RAPP APIs use *application/x-www-urlencoded* forms when only a JSON tree is to be sent and *multipart/form-data* to send binary data.

Access rights to the RAPP Platform services are being handled by an authentication server, hosted on the platform. The authentication server has been developed to deliver token-based application authentication solutions. Custom authentication http header implementation is used to deliver application tokens to the Platform. When a robot platform connects to the Platform for the first time, it has to invoke the service responsible for creating application tokens. The user/robot token acquired from the RAPP Store on registration is required in this phase. That token is used to authenticate existence of the user/robot and obtain information on service access rights. Next, application tokens are created, by the authentication server, stored under the platform's database and returned to the robot. At this point, the Core Agent is responsible of storing the application tokens on the robot for future usage. An application token can give access to more than one platform services/applications.

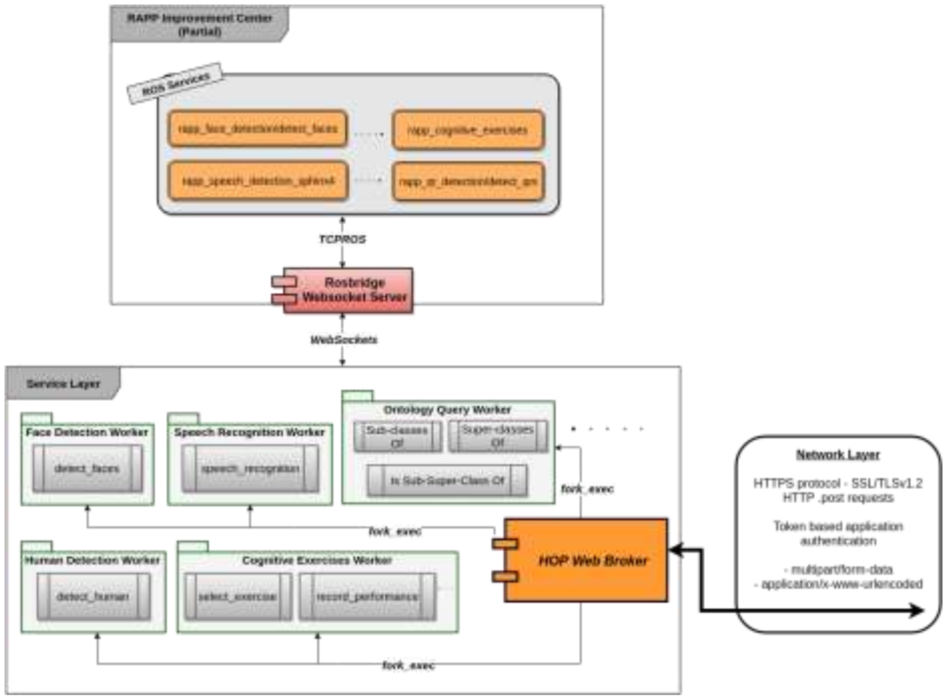

Fig. 3. The RAPP Platform communication layer, developed in HOP and responsible for handling all external communications to and from the RAPP Platform via web sockets.

On each client request to the Platform, a connection to the authentication server is established in order to authenticate the client/robot against the requested service, using the token that has been sent on the current request. Upon successful authentication, proper ROS interfaces are automatically created/instantiated thus enabling the communication server's services to "talk ROS", through the rosbridge web socket transport layer. In case authentication is not successful, an HTTP 401 Unauthorized response is delivered to the client. Figure 4 presents the aforementioned authentication mechanism used to authenticate robot platforms for accessing RAPP Platform's services.

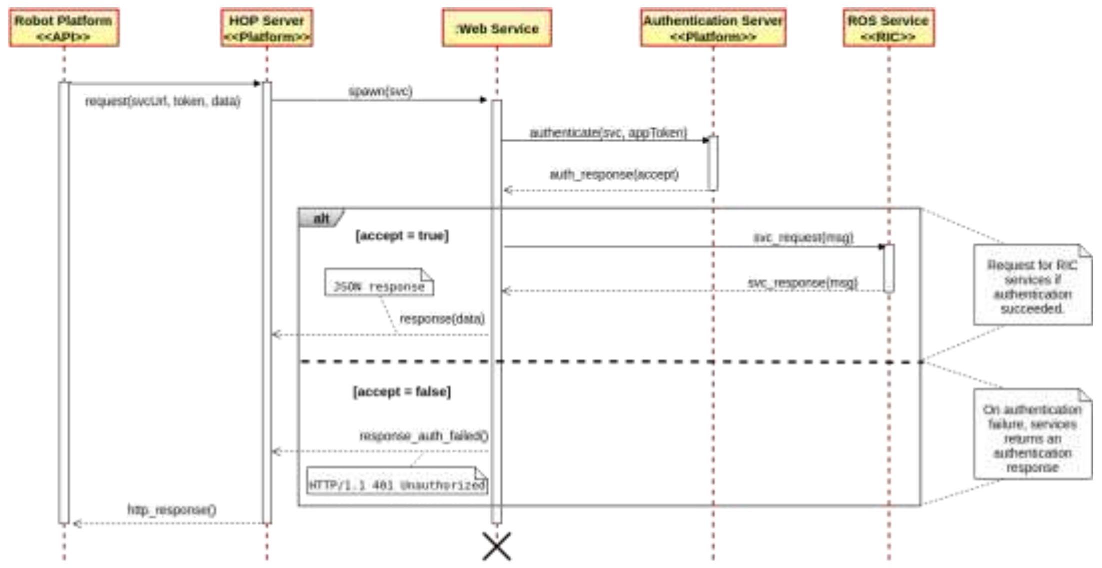

Fig. 4. An illustration of the RAPP authentication mechanism.

3.6. The RAPP Cloud agents

A Cloud Agent features a common execution workspace where any RApp developer can offload computationally expensive processes not already supported by the RIC's services. Any developer can in practice program custom made processes and services that will be executed in an isolated workspace and provide support functionalities to the RApps.

In order for a developer to deploy his own Cloud Agent several steps are required. Initially, they must submit the Cloud Agent to the RAPP Store. Next, the necessary files are submitted to a specific RAPP Platform node responsible for handling RAPP Cloud Agents, which in turn builds and deploys the Cloud Agent following the procedure and specifications presented below.

For each RApp containing a Cloud Agent, a dedicated Docker[§§] container will be available, isolating the RApp from the hosting environment. The RApp services will be available through the HOP Server for any Dynamic Agent to invoke. In order for the Cloud Agent to be properly deployed and utilized the RApp developer must provide the Cloud Agent code along with some utility files, which include a setup script, a deployment script and a parameters file. The setup script will be executed inside the container installing the needed Cloud Agent's dependencies, as well as the Agent himself. After the installation is complete, the deployment script will deploy the Cloud Agent. The configuration file will contain information regarding the services the Cloud Agent provides, i.e. the name of the service provided, the port which the service will use in order to communicate and the type of the service provided, i.e. ROS, C++ etc.

[§§] https://www.docker.com/

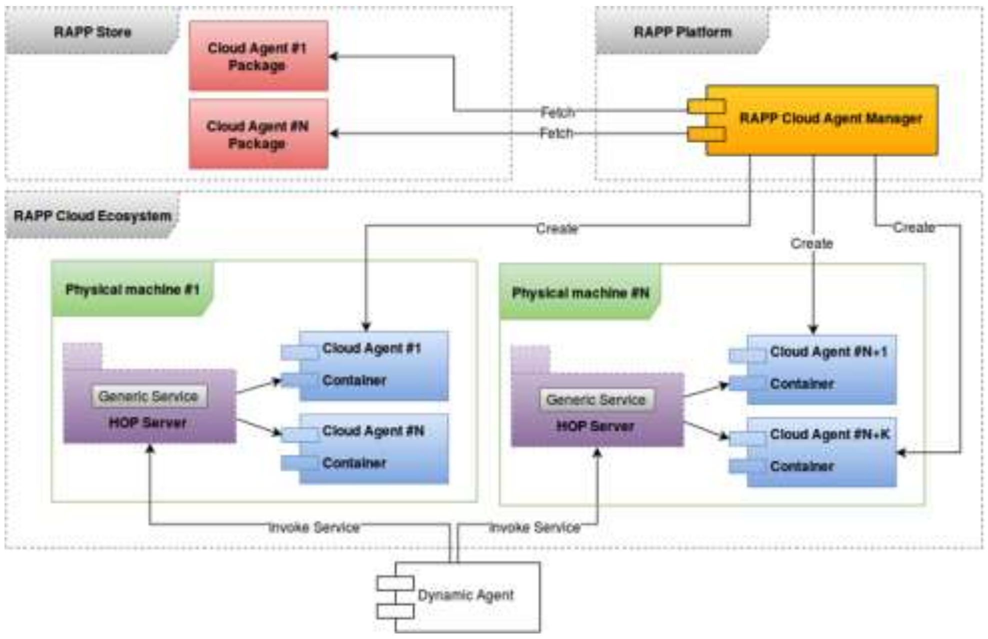

Fig. 5. The architecture of the RAPP Cloud agent subsystem along with its dependency to the RAPP Platform Cloud Agent Manager and its communication with the external Dynamic Agent.

This way, the developer is able to fully parameterize the dedicated container towards automatically deploying their Cloud Agent. However, he is responsible for implementing the needed functionality to read the service arguments provided by the HOP Server. Since no restrictions exist in the Cloud Agent provided services template, all services must share the same generalized format. This way the HOP handlers and the HOP service callers (ROS & Named sockets) will be static.

For each physical machine a single HOP Server will be deployed that will be responsible of serving the requests to the appropriate containers/Rapp Cloud Agents. The names and ports of the services will be available to the HOP Server via a parameter file and specifically the ports inside the container will be mapped to host ports during container deployment. The overall architecture of the RAPP Cloud agent is presented in Figure 5.

3.7. RAPP API, communication and security issues

The RAPP API, developed as part of RAPP project, acts as the bridge between all the involved entities and users. The overall concept is that the API will facilitate the cloud aspect of the RAPP Platform, by providing developers a way to implement RApps, e.g., cloud-robotic controllers. The RAPP API is responsible for integrating the subcomponents of the whole RAPP system. It allows for the cross-robot development of binaries, and provides a high-level abstraction of the base functions via a RAPP wrapper.

Two classes of RAPP API(s) exist. The first one is the *Platform API*, developed to deliver methods to invoke RAPP Platform’s cloud services. The second class is the *Robot API* for accessing robot-specific functionalities. Robot API(s) are considered to be per-robot and must exist for each robot platform supported by the RAPP ecosystem. Similar to current approaches in developing multi-threaded applications, one does not want to be burdened with the specifics of low level threads; instead a developer should be using the RAPP API to simply invoke high level abstracted function and classes in order to effortlessly create a RApp. The API provides a significant level of abstraction to the developers, nevertheless, they should be aware of which components will ultimately run on the robot and which on the cloud.

Here a special mention must be made concerning the development of the RAPP Platform API and more importantly of the RAPP Robot API. As described, a RApp (Robotic Application) should utilize at least the RAPP Robot API in order to manipulate the sensors and effectors of the robot. Furthermore, if the RApp developer does not desire (or is not able) to implement higher level robotic algorithms, or if the robot's resources are prohibiting the execution of computationally intensive modules, the RAPP Platform API can be used as well, with or without the support of Cloud agents. Finally, an advanced developer can potentially utilize all the above in conjunction with external libraries and tools, such as ROS, OpenCV etc. Nevertheless, in the case where a RApp only utilizes the RAPP Robot and Platform APIs, and taking under consideration the assumption that all supported robots have Unix as their OS, it can be stated that these applications are cross-robot or robot-agnostic in nature. This is supported by the development approach of the RAPP Robot API, performed using the ABC (Abstract Base Classes) paradigm. This way, the RApps only utilize the common, abstract API calls, whereas each robot's specific implementation is delivered along with the core agent. Of course, a robot-agnostic application makes sense when robots with similar abilities are concerned. For example, if an application performs obstacle avoidance, any robotic vehicle able to move and acquire distance sensory information will be able to execute this RApp (e.g. a wheeled robot or NAO). On the other hand, if a robot does not possess some of the aforementioned abilities, the calls will simply have no effect, potentially resulting to unforeseeable behavior. For this reason, a short description of the expected capabilities a robot must have for a RApp to be executed must be provided along with the application itself.

We currently provide and maintain Platform APIs, in C++, JavaScript and Python programming languages. These have been developed to support HTTP persistent connections to the Platform according to RFC-7230[***]. Benefits of using http persistent connections are the avoidance of slow connection setups and faster data transfers. The communication server of the platform accepts only https connection requests. The protocol used for https connections is the Transport Layer Security Protocol Version 1.2 (TLSv1.2) as specified in RFC-5246[†††], in order to ensure privacy and data integrity between the clients (robot platforms) and the platform. Proper TLSv1.2 adapters have been integrated into each Platform API implementation (C++, JavaScript, Python).

3.8. RAPP Security and Data Privacy

To address data privacy and security concerns, RAPP builds upon secure data transmission protocols. Communications taking place between the RAPP Platform and the robots are authenticated and encrypted over the HTTPS protocol. User specific data is stored in the ontology knowledge management system anonymously. The data is tied to the users by a random string of characters and numbers, denoted as the user's ontology alias. The link that ties a user's name and personal information to his ontology alias, is stored in a MySQL database, separate from the ontology knowledge management system and inaccessible from the

[***] https://tools.ietf.org/html/rfc7230

[†††] https://tools.ietf.org/html/rfc5246

outside world over the internet. RApps are explicitly denied access to it, thus preventing potential malicious activity. The database can be accessed only by predefined services running on the RAPP Platform which were developed as part of the RAPP System architecture and are shielded against malicious attempts to hack the database (SQL injections) according to up-to-date and established software practices. The whole RAPP Platform and RAPP Store are hosted on a private cloud infrastructure used solely for the purposes of the RAPP Project. No other application or system is hosted on the same infrastructure and no other user, entity or company not involved in the RAPP Project can access it. Sensitive user data can only be accessed by those having physical access to the infrastructure where RAPP is hosted.

On the robot side, RApps are running in sandboxed mode, unable to intervene with the execution of other RApps running simultaneously. In addition, as RApps cannot access personalized data, it would be impossible to acquire such information by tampering with the execution of an active RApp. Theoretically, it is possible that a malicious developer creates a RApp that undertakes behavior undesired by the user (monitoring him, for example) without his knowledge or consent. This issue however is not specific to the RAPP project as in practice any application currently existing in online smartphone application stores could undertake such action. To prevent such action, we plan to review all RApps submitted to the RAPP store when it reaches a production state. The only valid mechanism to ensure proper application activity is to analyze the source code, an approach followed by all major application stores. Prior to downloading and using a RAPP, users will be notified whether it has been reviewed and deemed safe or not and will then be prompted to act accordingly. We believe that older people, especially those with cognitive impairment should be prevented from using RApps that have not reviewed and deemed safe, thus a lock mechanism exists that will prevent them from downloading and using them. The review process will also include quality control enforcing technical as well as usability standards in order to enhance the user's experience. An in-house pre-release review will take place for "advanced" or "commercial" developers, where we will provide an extra level of quality control. RApps will also be able to be reviewed and voted by users and professionals through the RAPP Store.

We are confident with the data privacy and security measures in place in order to prevent unauthorized access to sensitive user data. Naturally, limitations may exist in our approach, but they are equivalent to those of most online public services in our society. Regarding privacy issues, we can guarantee that user data will not be used for purposes not stated in our agreement with the user and beyond the scope of goals of the RAPP project.

4. RAPP Ecosystem evaluation

In order to establish the ability of the RAPP framework to support the creation of quality RApps, the RAPP approach from both developers and RApp users perspectives was evaluated. Regarding the robot user's perspective, focus groups of end users that have been collaborating with the development team upon the initialization of the development have been engaged. These focus groups have described and analyzed their desired requirements regarding robotic applications and have tested and evaluated the produced RApps providing their feedback. During the development and implementation of the first RApps several application requirements have been specified and applied, both in the RAPP Platform and the RAPP API.

Two robotic platforms are included into the RAPP development cycle: The Aldebaran NAO humanoid[‡‡‡] robot, and INRIA's ANG-MED[§§§] robot. The use of two heterogeneous robots, addressing completely different needs, serves to demonstrate the platform’s interoperability and provides evidence of the generalization of the RAPP concept by wrapping different capabilities through the RAPP API.

An initial evaluation of the RAPP ecosystem was performed through three distinct pilot cases, each targeting a different user group. The pilot cases involve the development of RApps for the NAO humanoid robot and the ANG-MED rollator robot, targeting senior citizens that are either a) technology illiterate, b) have been diagnosed with mild cognitive impairment or c) are in the process of hip fracture rehabilitation. In the following sections, the outcomes and insights of the three pilot cases will be described.

Regarding the developer’s perspective, a technical programming event was organized for developing RApps utilizing the RAPP framework, where more than twenty (20) external developers participated and provided feedback.

4.1. MCI and technology illiterate user groups

Seniors citizens from the local Community Centre at New Moudania in Greece participated in this pilot testing RAPP scenario of technology illiterate people interacting with a robot, NAO in this case. According to the initial specifications provided by the focus group, the RApps developed for this user group must be able to deliver user friendly applications that will enable older adults to effortlessly use them with simple actions. Execution via voice commands must be supported as well as the ability to recognize the user and salute him using his name in order to create a sense of intimacy between them. The RApps must also enable users to communicate with family and friends through emails in a simple way and by specific oral commands so as to allow them to maintain their social connections and social inclusion. Lastly, the RApps must engage the user in cognitive games available through the robot that aim to enhance her attention and memory.

A focus group of six (6) technology illiterate older adults (65+) have tested the Email-Handler and Cognitive exercise RApps via the NAO robot. For speech recognition/detection purposes, both applications use the Speech Detection Platform services. Furthermore, the applications utilize a series of Platform services and NAO functionalities through the Platform and Robot APIs respectively.

As the development of usable robotic applications for older adults is still a challenge, we have discussed with the focus group the perceived usability of the tested RApps. It has to be mentioned here that the focus group of technology illiterate users who tested the initial applications was at this point technology literate and possessed the needed experience to compare and evaluate applications produced by computers or robots.

Moreover, the set of RApps developed in Greek language was evaluated by five professionals working with older people diagnosed with Mild Cognitive Impairment (three psychologists, one social worker and one physical trainer). According to their opinion, the behavior of the prototype RApps delivered via NAO has been classed as impeccable. Success rate on recognizing Greek words was high under low-noise or no noise

[‡‡‡] http://www.aldebaran.com/en/humanoid-robot/nao-robot

[§§§] https://pal.inria.fr/research/themes/rehabilitation-transfer-and-assistance-in-walking/walking-aids/

environments. In order to better capture the responses from the users, the de-noise profile configuration was addressed dynamically.

The technology illiterate group of users, as well as the professionals working with older adults diagnosed with MCI, pointed out that RApps must be able to motivate users to utilize them. They must involve simple steps in completing a task that does not include procedures requiring memorization such as passwords or menu paths and communicate clear messages to the user. Furthermore, they must be personalized and adjusted to the needs of the user. Lastly, an adequate sound system is critical and especially the robotic voice must be clear, nice and easy to understand.

As usability is defined by the interaction between the user and the system and can be evaluated according to the performance of the users, their satisfaction from the product or system and its acceptability (Bevan, 1995), initial testing has already demonstrated that RApps successfully address these principles. Observations and interviews (both personal and group) with the initial user group of technology illiterate older adults have revealed that RApps were pleasurable and satisfying. The users enjoyed to interact with the robot towards which they felt very familiar after their first meeting. Their experience on using the applications was exceptional as they found them built in simple and easy steps.

4.2. Mobility issues user group

The goals of an assistive device to the rehabilitation process is to improve independent mobility, reduce disability, delay functional decline, decrease the burden of care and provide assessment tools for the medical community (Bateni & Maki, 2005; Faruqui & Jaeblon, 2010). Patients using assistive devices have reported improved confidence and feeling of safety resulting in increased activity levels and independence (Bradley & Hernandez, 2011; Smith, Quine, Anderson, & Black, 2002). Assistive devices such as canes, crutches, and walkers can be used to increase a patient’s base of support, improve balance and increase activity and independence without interfering with the rehabilitation outcomes (Lucki & Bach, 2010). RAPP applies the most innovative solutions to provide support for people with mobility problems and who are under a rehabilitation process. Specially developed RApps identify and personalize the provided support according to each user’s features and needs while also motivating them to improve their activity level and perform exercises adapted to their physical status. Moreover, RApps can monitor the walking parameters of the patient to analyze and report his walking pattern and any subsequent improvements to his professional therapist.

The complete setup consists of the ANG-MED robot, an infrared remote used to directly control the robot, a PC control station used by the caregiver to operate the robot and access activity data and the RAPP platform which hosts the RAPP store, the remote applications and the platform services used by these applications (in our case, the authentication/authorization services, the database of activity data, and the server application for the PC control station). Users wear optional RFID wristbands for automatic identification. Currently there is no standard for robotized mobility assistance and it has not been commercially deployed yet.

Mobility assistance and mobility monitoring RApps were tested with the ANG-med robot in a lab environment at the Matia Gerontological Institute. The researchers of the Matia Institute (n=6) and care professionals (n=10) of the Bermingham Hospital have tested the developed applications.

Hardware and firmware tests validated the ability to perform the exercises defined by MATIA by operating the ANG-med actuators, assess data and synthesize relevant indicators. Tests on the RApp applications demonstrate the viability of the distributed software architecture of the RAPP project. Web user interfaces have been developed for caregivers and patients enabling them to provide feedback.

RApps have been further tested at MATIA's lab in order to assess functionalities and already installed exercises. Usability criteria have been taken into account rather than clinical criteria. The functionalities tested via the ANG-med prototype by INRIA at their facilities were 10 minutes walking, timed up and go (TUG), inverted L exercise and maze. The functionalities tested at Matia's lab were identification, dynamic walk exercise, correct position and form, timed up and go, hip extension and hip flexion.

The first results advanced the possible benefits of using these RApps for rehabilitation purposes, as the system managed to recognize the wrong postures and practices and corrected the position and moves of the users. Minor monitoring issues were also addressed.

4.3. Evaluation from the perspective of developers

The creation of quality RApps is one of the most important aspects in addressing the requirements and needs of the older people using RAPP's robots. The goal is for external developers to become actively involved in RAPP and spark a RApp creation interest. In order to evaluate the usability of the RAPP framework a RApp creation event was organized, aiming at bringing RAPP and the developing community closer. Our motivation was to record the needs of developers, adapt to their practices and address their concerns.

Twenty (20) external developers participated, most of which were graduate students, along with some professionals. After a short, one-hour introductory course to the RAPP project and the RAPP API followed a 2-hour development effort where the participants either formed teams or worked individually. RAPP technical team members were present, in order to assist in the development processes. Development was performed solely in Python in an Ubuntu Linux based environment. Participant experience in Python and Linux varied significantly with most of them having average experience.

Nevertheless, their lack of significant experience was not an obstacle towards creating interesting and functional RApps. The usability and the high-level abstraction of the RAPP framework allowed for accelerated development and fast prototyping of ideas to practical applications. Three (3) NAO robots were available for testing and live demonstration purposes. Developers were thrilled to see their application executed in the actual robot rather than some computer based simulation. Additionally, live demonstration of applications attracted attention and promoted interaction and idea sharing between development teams.

The number of RApps developed, along with their complexity and depth, were inspiring, especially when accounting for the short developing timeframe of 2 hours. Some of the most interesting applications created in the context of the event were English to Greek translation via RAPP Speech recognition and the external Google translate API, 2D mapping via rotational motion and sonar feedback, operating NAO as a music player by storing songs in-robot and playing them back via voice commands, face detection and recognition via OpenCV and lastly, a gesture application where NAO showcased complex motor skills.

Developer feedback regarding usability, usefulness and adequacy of features was dominantly positive as illustrated in Figure 6, a fact that inspires us towards further improving our software infrastructure.

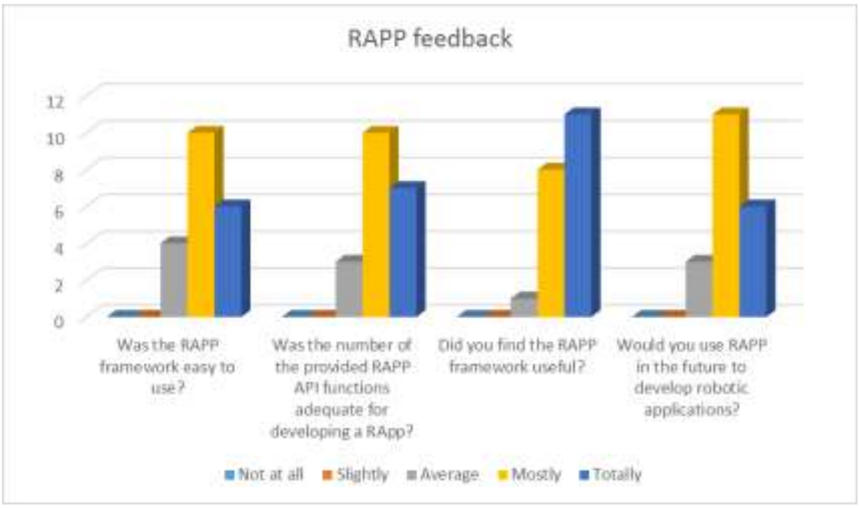

Fig. 6. Developer feedback regarding the usability, ease of use and plethora of provided features functions of the RAPP framework.

All developers found the RAPP framework interesting and straightforward and were eager to participate in our future events. More than 90% had enough time to develop an application and would be interested in developing RApp applications for other robots as well. In the immediate future, developer feedback has geared us towards expanding the motion functions for the NAO robot, incorporating support for new robots and expanding the RAPP Platform with capabilities like motion detection and OCR.

5. Discussion and future steps

The users' feedback disclosed their experience which was in general positive and expressed their satisfaction with the RApps which were generally characterized as simple to use, motivational, personalized and adjusted to their needs. The e-mail Handler RApp facilitated the communication of the users with family and friends in an effortlessly simple mode. Cognitive games were interesting, targeting the working memory of the users and trained their attention skills. Moreover, their numerous variations and different difficulty levels motivated users to improve their performance while remaining an enjoyable activity. On the other hand, the Mobility assistance and monitoring RApps enabled users to perform their daily rehabilitation exercises in the right manner by correcting their position and movements according to the RApp feedback (via the rollator) with minimum intervention by a caregiver.

The RApps evaluation, apart from revealing the benefits of the applications to users and their effect on their quality of life, has revealed some weaknesses that were immediately addressed and corrected. More RApps are currently under development, mostly targeted towards use cases involving the NAO robot. As the supporting RAPP platform services are mature enough, we are currently developing the actual RApps on the robot side and working on optimal content delivery to the end user. As for the ANG-med, a second prototype is being developed taking into account the feedback and outcomes of the testing performed at a lab environment. This second prototype will be tested for 6 months in a real world hospital environment by both patients and professionals.

We plan to test both current and the under development RApps in a large scale environment where groups of healthy older adults along with older adults diagnosed with MCI and caregivers will be involved. Apart from the qualitative method analysis which up to now has served well the research needs, questionnaires are to be distributed to users, caregivers and professionals in order to collect statistical data for the usability of the applications and the human to robot interaction. The extended testing and evaluation that will follow is expected to be constructive and fruitful in enriching the platform with more functionalities and improved applications.

Feedback from developers has been dominantly positive while also constructive, allowing us to draw new insights and focus our attention on further streamlining the RApp creation process. We are in the process of organizing a larger hackathon event where more external developers will participate and engage in advanced RApp creation. We are actively addressing the developer community on every opportunity as their feedback is vital to our efforts of enhancing the usability of the RApp platform.

Concerning RAPP's technical aspirations, it should be stated that RAPP project's final goal is not to integrate all (or a large number of) robotic tasks required by the inclusion scenarios, but to provide a fully functional and extendable cloud robotics ecosystem. Even though we focus on the three described pilot cases (MCI elders, technologically illiterate elders and old people suffering from hip fracture), a whole infrastructure is provided to external developers, in order to create and distribute their own applications for any domain. Furthermore, special care is being taken for the RAPP Framework to be easy in terms of utilization for non-roboticists, or even for novice programmers.

Several pitfalls or negative aspects of RAPP in general may appear upon its final release, when a larger number of users is expected to utilize it. The first group RAPP targets is the developers who will directly utilize the core of the RAPP Framework, i.e. the RAPP APIs and the four distributed agents. One of the most common issues a developer may find frustrating, is the poor documentation of the provided software, either when detailed descriptions are missing or when the descriptions are misleading regarding a modules functionality.

In RAPP, a lot of effort has been put into creating correct documentation for all the software, as well as performing field tests to support this assumption by organizing technical hackathons. Other drawbacks involve limited amount of API calls (either in the robot or the platform APIs), no supported programming languages (other than the currently three supported – C++, Python and JS) and malfunctioning or complex packaging and distribution methods (i.e. the RAPP Store). The second directly affected user group includes the final users, them being the elders, their families or the caregivers. One possible discouragement regarding these groups towards utilizing the RAPP ecosystem is a possibly limited amount of available applications existent in the RAPP Store, fact that constitutes RAPP weak for doing their jobs. Finally, since these users are not technical in nature, one of the most disappointingly events that may happen is to download an application that malfunctions or does not start at all. This is usually one of the reasons for users (even with technical education) to quickly abandon any software in search for an alternative.

Finally, one of the concerns in further growth of RAPP, and especially of the RAPP store, is of course acceptance by the market, since RAPP aims to be established in the market of robotics apps stores. Same way as end-users of app stores are able to choose from large varieties of applications that can satisfy existing needs or cover totally new and unexpected types of needs, the RAPP store aims to provide this capability to robots that can choose, from a wide variety of RApps, the ones that fit them and the needs of their human companions best, according to a set of parameters like human companion profile, expressed needs, context of use, etc. At the moment there are only a few key players in this nascent robotic applications store market. These currently existing app stores have a limited number of applications available and support a limited number of robotic platforms. Furthermore, they don't offer cloud support for their apps, or utilise the cloud to offer advanced capabilities.

On the contrary, RAPP is offering apps for a variety of robots and platforms. What is more, RAPP is pioneering the concept of cloud robotics, offering apps that make use of cloud services to fulfil the needs of the users. And already having an alpha/beta version and experimenting with it allows us to create content which is very important. So this is a brand-new technology on which RAPP already has an early R&D advantage. Should we proceed swiftly to develop a commercially available platform, RAPP content and store, RAPP could become a leader in this robotics market. Being pioneers in the market means that RAPP has the potential to make waves and attract media attention. This is a first in the field opportunity. Still, the current robotic apps market is minuscule: only those with a humanoid robot (NAO, Jimmy, Pepper, etc) may currently take interest in RAPP content. Thus it is important to analyze the RAPP potential customers/buyers in order to examine both the technological requirements, and also where the potential revenue may come from.

6. Conclusion

In this paper, a novel integrated robotics architecture is proposed, targeting the needs of senior citizens at risk of social exclusion. The proposed architecture introduces a common API for developers to create robot agnostic applications while also providing advanced of-the-shelf services hosted in a cloud based-platform. In this way, RAPP provides developers the necessary tools and APIs to build their own assistive robotic applications, in order to improve the quality of life of people at risk of exclusion. The RApps will be hosted in a dedicated store, which is expected to ultimately have a profound effect in the robotic application market. In the context of the RAPP pilot cases, RApps have already been developed assisting older people in their daily activities, allowing them to communicate with friends and family, keeping them informed about regional and international news and weather conditions, practicing their cognitive skills with games that exercise attention and memory and supporting them in the rehabilitation process following a hip fracture. Early results have demonstrated the viability of the RAPP approach, its ability to effectively address the needs of both developers and end users and established its merit as a valuable tool in caring for older people.

Acknowledgements

Parts of this work was supported by the FP7 Collaborative Project RAPP (Grant Agreement No 610947), funded by the European Commission.

References

Bateni, H., & Maki, B. E. (2005). Assistive devices for balance and mobility: benefits, demands, and adverse consequences. *Archives of physical medicine and rehabilitation, 86*(1), 134-145.

Beetz, M., Mösenlechner, L., & Tenorth, M. (2010). *CRAM—A Cognitive Robot Abstract Machine for everyday manipulation in human environments.* Paper presented at the Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on.

Bemelmans, R., Gelderblom, G. J., Jonker, P., & De Witte, L. (2012). Socially assistive robots in elderly care: A systematic review into effects and effectiveness. *Journal of the American Medical Directors Association, 13*(2), 114-120. e111.

Bevan, N. (1995). Usability is quality of use. *Advances in Human Factors/Ergonomics, 20,* 349-354.

Bradley, S. M., & Hernandez, C. R. (2011). Geriatric assistive devices. *American family physician, 84*(4).

Bruyninckx, H., Klotzbücher, M., Hochgeschwender, N., Kraetzschmar, G., Gherardi, L., & Brugali, D. (2013). *The BRICS component model: a model-based development paradigm for complex robotics software systems.* Paper presented at the Proceedings of the 28th Annual ACM Symposium on Applied Computing.

Dudek, W., Szynkiewicz, W., & Winiarski, T. (2016). Nao Robot Navigation System Structure Development in an Agent-Based Architecture of the RAPP Platform *Challenges in Automation, Robotics and Measurement Techniques* (pp. 623-633): Springer.

Dutoit, T., Pagel, V., Pierret, N., Bataille, F., & Van der Vrecken, O. (1996). *The MBROLA project: Towards a set of high quality speech synthesizers free of use for non commercial purposes.* Paper presented at the Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on.

Elkady, A., Joy, J., & Sobh, T. (2011). *A plug and play middleware for sensory modules, actuation platforms and task descriptions in robotic manipulation platforms.* Paper presented at the ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

Faruqui, S. R., & Jaeblon, T. (2010). Ambulatory assistive devices in orthopaedics: uses and modifications. *Journal of the American Academy of Orthopaedic Surgeons, 18*(1), 41-50.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on, 32*(9), 1627-1645.

Gerkey, B., & Garage, W. (2009). *Towards Robot App Store.* Paper presented at the International Joint Conference on Artificial Intelligence (IJCAI) Robotics Workshop.

Ha, Y.-G., Sohn, J.-C., Cho, Y.-J., & Yoon, H. (2005). Towards ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework. *ETRI journal, 27*(6), 666-676.

Hunziker, D., Gajamohan, M., Waibel, M., & D'Andrea, R. (2013). *Rapyuta: The roboearth cloud engine.* Paper presented at the Robotics and Automation (ICRA), 2013 IEEE International Conference on.

Hutson, S., Lim, S. L., Bentley, P. J., Bianchi-Berthouze, N., & Bowling, A. (2011). Investigating the suitability of social robots for the wellbeing of the elderly *Affective computing and intelligent interaction* (pp. 578-587): Springer.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., . . . Darrell, T. (2014). *Caffe: Convolutional architecture for fast feature embedding.* Paper presented at the Proceedings of the ACM International Conference on Multimedia.

Joyeux, S., & Albiez, J. (2011). *Robot development: from components to systems.* Paper presented at the 6th National Conference on Control Architectures of Robots.

Kintsakis, A. M., Reppou, S. E., Karagiannis, G. T., & Mitkas, P. A. (2015). *Robot-assisted cognitive exercise in mild cognitive impairment patients: The RAPP approach.* Paper presented at the E-Health and Bioengineering Conference (EHB), 2015.

Lucki, K., & Bach, M. (2010). Rollator use and functional outcome of geriatric rehabilitation. *Journal of rehabilitation research and development, 47*(2), 151.

Lutz, W., Sanderson, W., & Scherbov, S. (2008). The coming acceleration of global population ageing. *Nature, 451*(7179), 716-719.

Mitkas, P. A. (2015). Assistive Robots as Future Caregivers: The RAPP Approach *Progress in Automation, Robotics and Measuring Techniques* (pp. 171-179): Springer.

Nielsen, S. H., Bøgild, A., Jensen, K., & Bertelsen, K. K. (2011). *Implementations of frobomind using the* *robot operating system framework.* Paper presented at the NJF Seminar 441 Automation and System Technology in Plant Production.

Osentoski, S., Jay, G., Crick, C., Pitzer, B., DuHadway, C., & Jenkins, O. C. (2011). *Robots as web services: Reproducible experimentation and application development using rosjs.* Paper presented at the Robotics and Automation (ICRA), 2011 IEEE International Conference on.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., . . . Ng, A. Y. (2009). *ROS: an open-source Robot Operating System.* Paper presented at the ICRA workshop on open source software.

Roalter, L., Kranz, M., & Möller, A. (2010). A middleware for intelligent environments and the internet of things *Ubiquitous Intelligence and Computing* (pp. 267-281): Springer.

Saffiotti, A., Broxvall, M., Gritti, M., LeBlanc, K., Lundh, R., Rashid, J., . . . Cho, Y.-J. (2008). *The PEIS-ecology project: vision and results.* Paper presented at the Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.

Sanfeliu, A., Hagita, N., & Saffiotti, A. (2008). Network robot systems. *Robotics and Autonomous Systems, 56*(10), 793-797.

Schlegel, C., Haßler, T., Lotz, A., & Steck, A. (2009). *Robotic software systems: From code-driven to model-driven designs.* Paper presented at the Advanced Robotics, 2009. ICAR 2009. International Conference on.

Serrano, M., & Berry, G. (2012). Multitier programming in hop. *Communications of the ACM, 55*(8), 53-59.

Serrano, M., Gallesio, E., & Loitsch, F. (2006). *Hop: a language for programming the web 2. 0.* Paper presented at the OOPSLA Companion.

Serrano, M., & Queinnec, C. (2010). A multi-tier semantics for Hop. *Higher-Order and Symbolic Computation, 23*(4), 409-431.

Smith, R., Quine, S., Anderson, J., & Black, K. (2002). Assistive devices: self-reported use by older people in Victoria. *Australian Health Review, 25*(4), 169-177.

Szlenk, M., Zieliński, C., Figat, M., & Kornuta, T. (2015). Reconfigurable Agent Architecture for Robots Utilising Cloud Computing *Progress in Automation, Robotics and Measuring Techniques* (pp. 253-264): Springer.

Tenorth, M., & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. *The International Journal of Robotics Research, 32*(5), 566-590.

Tenorth, M., Klank, U., Pangercic, D., & Beetz, M. (2011). Web-enabled robots. *Robotics & Automation Magazine, IEEE, 18*(2), 58-68.

Tenorth, M., Nyga, D., & Beetz, M. (2010). *Understanding and executing instructions for everyday manipulation tasks from the world wide web.* Paper presented at the Robotics and Automation (ICRA), 2010 IEEE International Conference on.

Waibel, M., Beetz, M., Civera, J., d'Andrea, R., Elfring, J., Galvez-Lopez, D., . . . Perzylo, A. (2011). A World Wide Web for Robots≫. *IEEE Robotics & Automation Magazine, 18*(2), 69-82.

Walker, W., Lamere, P., Kwok, P., Raj, B., Singh, R., Gouvea, E., . . . Woelfel, J. (2004). Sphinx-4: A flexible open source framework for speech recognition.

Wang, W., Johnston, B., & Williams, M.-A. (2012). Social networking for robots to share knowledge, skills and know-how *Social Robotics* (pp. 418-427): Springer.

Wolf, P., Schmidt, A., Otte, J. P., Klein, M., Rollwage, S., König-Ries, B., . . . Gabdulkhakova, A. (2010). *openAAL-the open source middleware for ambient-assisted living (AAL).* Paper presented at the AALIANCE conference, Malaga, Spain.

Wurm, K. M., Hornung, A., Bennewitz, M., Stachniss, C., & Burgard, W. (2010). *OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems.* Paper presented at the Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation.

Zander, S., Heppner, G., Neugschwandtner, G., Awad, R., Essinger, M., & Ahmed, N. (2016). A Model-Driven Engineering Approach for ROS using Ontological Semantics. *arXiv preprint* *arXiv:1601.03998*.

Zielinski, C., Szynkiewicz, W., Figat, M., Szlenk, M., Kornuta, T., Kasprzak, W., . . . Figat, J. (2015). *Reconfigurable control architecture for exploratory robots.* Paper presented at the Robot Motion and Control (RoMoCo), 2015 10th International Workshop on.