HAL
archives-ouvertes.fr

RAPP System Architecture

Fotis Psomopoulos, Emmanouil Tsardoulias, Alexandros Giokas, Cezary Zielinski, Vincent Prunet, Ilias Trochidis, David Daney, Manuel Serrano, Ludovic Courtès, Stratos Arampatzis, et al.

▶ **To cite this version:**

Fotis Psomopoulos, Emmanouil Tsardoulias, Alexandros Giokas, Cezary Zielinski, Vincent Prunet, et al.. RAPP System Architecture. Assistance and Service Robotics in a Human Environment, IEEE/RSJ International Conference on Intelligent Robots and Systems, Sep 2014, chicago, United States. <hal-01090891>

HAL Id: hal-01090891
https://hal.inria.fr/hal-01090891

Submitted on 4 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RAPP System Architecture

Fotis Psomopoulos, Emmanouil Tsardoulias, Alexandros Giokas, Cezary Zielinski, Vincent Prunet, Ilias Trochidis, David Daney, Manuel Serrano, Ludovic Courtes, Stratos Arampatzis, Pericles A. Mitkas

***Abstract*— Robots are fast becoming a part of everyday life. This rise can be evidenced both through the public news and announcements, as well as in recent literature in the robotics scientific communities. This expanding development requires new paradigms in producing the necessary software to allow for the users' particular needs. In this paper we present a novel architectural design of the RAPP framework that attempts to address this issue, developed within the context of the EU funded project RAPP "Robotic Applications for Delivering Smart User Empowering Application". The proposed framework has been designed aiming towards a cloud-based approach to integrating robotic devices and their respective applications. This goal was defined going beyond the upcoming trends in infrastructures, and focusing on alternative approaches to conventional robotic controllers, while at the same time expanding the capabilities of the RAPP framework in a seamless and scaling manner.**

I. INTRODUCTION

It is becoming increasingly evident that our current social infrastructures and services are struggling to keep up with the dramatic demographic changes apparent in our societies. It is a well known reality, that in the near future, elderly and people requiring support in their daily life will increase and caregivers will not be enough to assist and support them [1]. Socially interactive robots can help to ameliorate this situation, not only by physically assisting people, but also by functioning as a companion [2]. The increasing sales figures of robots are indicating that we are witnessing a rising trend for social robotics [3][4][5][6]. In order to lower the cost for developers and to increase their interest on developing robotic applications, the RAPP project introduces the idea of robots as platforms.

The project, aptly named "Robotic Applications for Delivering Smart User Empowering Applications" (RAPP), aims to provide an open-source software platform to support the creation and delivery of robotics applications (RApps). At the same time, and by utilising our platform, the first implemented RApps are targeting people at risk of exclusion, especially older people. By relying on our platform, developers can utilise the respective Application Programming Interface (API) that is provided, which contains the functionalities for implementing RApps, as well as accessing the robot's sensors and actuators using higher level commands. This is achieved by inserting a middleware stack with added functionalities suitable for different kinds of robots. RAPP expands the computational and storage capabilities of robots and enables machine learning operations, distributed data collection and processing, and knowledge sharing among robots in order to provide personalised applications, based on adaptation to individuals. The use of a common API, assists developers in creating improved applications for different types of robots, whilst addressing people with different needs, capabilities and expectations, and at the same time respecting their privacy and autonomy. Thus, the proposed RAPP Store, will ultimately have a profound effect in the robotic application market.

Chapter II describes the state-of-the-art concerning specific examples of cloud robotic architectures, as well as online stores that host and distribute robotic applications. In chapter III the overall RAPP architecture is presented. Initially an overview is provided followed by a detailed description of the two main parts, i.e., the robot-side and the platform-side components. Additionally, special attention is paid to the different system users. Chapter IV is fully dedicated to the description of the proposed companion API, and finally, chapter V contains the conclusions as well as future possible extensions and planned work.

II. STATE OF THE ART

Because the RAPP project is a combination of robotic architecture, cloud robotics and application stores, a brief overview of the main representatives of each category is provided. Regarding robotic architectures, one of the key players is the Robotic Operating System (ROS), which is a framework and a middleware oriented towards writing robot software. It is a collection of tools, libraries and nomenclatures, which provide the necessary abstraction for a robotic developer to be able to create effortlessly complex robotic applications. It is described as a meta-operating system [7], as it provides standard system operating services, such as hardware abstraction, low-level device control, implementation of commonly used functionality and message-passing between processes and package management. It is fully distributed and asynchronous, as it allows for trans-

Fotis Psomopoulos, Emmanouil Tsardoulias (equal contributors) and Pericles A. Mitkas are with ITI - Information Technologies Institute, CERTH - Centre for Research and Technology Hellas, Thermi 57001, Greece fpsom@iti.gr, etsardou@iti.gr, mitkas@iti.gr

Alexandros Giokas, Ilias Trochidis and Stratos Arabatzis are with Ortelio Ltd Coventry University Technology Park Puma Way, Coventry, CV1 2TT UK a.gkiokas@ortelio.co.uk, it@ortelio.co.uk

Cezary Zielinski is with Institute of Control and Computation Engineering, Warsaw University of Technology C.Zielinski@elka.pw.edu.pl

Vincent Prunet and Manuel Serrano are with Inria Sophia Antipolis Mediterranee, 2004 route des Lucioles, F-06902 Sophia Antipolis cedex, France vincent.prunet@inria.fr, manuel.serrano@inria.fr

David Daney and Ludovic Courtes is with Inria Bordeaux-Sud Ouest, 200 rue vieille tour 33405 Talence, France ludovic.courtes@inria.fr

parent 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 that ROS is involved with. For example in [8], Joyeux, Sylvain, and Albiez investigate ROS 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 [9], allows it to be the link between intelligent environments and the "Internet Of Things" [10]. Elkady, Joy and Sobh [11], use ROS as a plug-and-play middleware for sensory modules, actuator platforms and task descriptions in robotic manipulation platforms, whereas in [12], Beetz et. al. use it as 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 standardisation of 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 become a low-level form of robotic application store, one where all apps are free and distributed through standard Linux package managers, such as apt-get [13][14].

There already exist a few classic application stores for robots, as robots slowly emerge as household devices. One of the most famous Stores is the RobotAppStore [15], 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 charge their products with a specific price. Another developer or end-user can then purchase, download and install the software on his or her robot. Another robot app-store paradigm is Aldebaran's NAO Store [16], which solely hosts applications that can be installed on the NAO robot. The main difference between NAO Store and RobotAppStore, apart from the target 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 employment. A major factor in the incorporation of robotic devices in our everyday lives, is the so called "Cloud Robotics", which essentially describes a world wide web for robots, a place where robots can collaborate by exchanging knowledge about their environments, contributing to the increase of the collective robotic cognition [17][18]. There are several projects that aim to establish a common knowledge pool for the entirety of robots, one of the most important being RoboEarth, an FP7-ICT project [19]. As RoboEarth's consortium 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, about their behaviour and their environment [20]. Additionally it employs KnowRob [21], a knowledge ontology database for robots. This is a knowledge processing system that combines knowledge representation and reasoning methods with techniques for acquiring knowledge and grounding the knowledge in a physical system and can serve as a common semantic framework for integrating information from different sources. Finally it support different deterministic and probabilistic reasoning mechanisms, clustering, classification and segmentation methods, and includes query interfaces as well as visualisation tools.

Another example of collective robot knowledge acquisition is displayed by the ROBOHOW.COG project, which is about 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, by 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 [22][23]. It must be stated that ROBOHOW uses KnowRob as well.

Other similar projects include PEIS-Ecology [24], which combines ideas from the field of autonomous robotics and ambient intelligence towards creating integrated house social robotic systems, URC (Ubiquitous Robotic Companion) [25], where the main idea is that robots have always access to services regardless of environment alterations and NRS (Networked Robot Systems) [26] which provides collaboration between physical robots, environment sensors / actuators and humans, with the help of autonomous network-based systems.

RAPP is different from the aforementioned technologies, by the fact that it tries to combine their functionalities into a single system implementation. In specific, RAPP system consists of a web (cloud) part, which contains the RApp-Store (Robotic Applications Store), as well as the knowledge pool and inference methodologies, and the robot-component that represent each robotic device that can download and install RApps. The cloud-part also provides services for employment of heavy duty computations on the web instead on the robot. The component that bridges together the entire system, is HOP, a toolset for programming the Web Of Things [27][28][29]. HOP is a multitier programming environment, built on top of web protocols and languages. It orchestrates data and commands 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 consist as an aggregation of communicating objects (sensors and active components such as robots) which are discovered, identified, and linked to client/server HOP software modules distributed among components. The different modules that comprise 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).

III. RAPP ARCHITECTURE

The proposed RAPP architecture has been designed as a distributed system, where the Platform and the Store are provided as cloud services, and the Client is located on the robot side. In that sense, the host is the controller executing on the robot, and the guest is the controller executing on the cloud. We consider both controllers as a singular entity, separated only by a network socket. Together, they make up the RAPP. In addition to the overall RAPP platform, we consider the RAPP Improvement Centre (RIC), also residing within the cloud component, which is in charge of performing the required Machine Learning (ML) and Data Mining (DM) processes, either upon request by a Robotic Application (RApp), or as a standalone process.

Each RApp residing on a robot, is either installed on demand or permanently stored on the robot, and is invoked by the RAPP Core Agent (described in detail in subsection IIIc). Moreover, any given RApp may also have a cloud counterpart. This is a module, provided by the RApp developer, which is to be executed on the cloud, beyond the hardware of the robot in question. Whilst the API provides that functionality, the ultimate decision of whether to employ such functionality lies with the developer. Finally, in order to address the involved security and privacy issues, any RApp executed on the cloud will be sand-boxed in a virtual machine and possibly encrypted.

A. Overall design

The overall RAPP architecture is depicted in Fig. (1), where it is apparent that RAPP comprises two main elements. The first part exists in the cloud and is the RAPP Platform, containing the necessary tools for hosting, distribution and execution of RApps, databases in which the overall system information is stored, as well as online and offline services that are available to the developers and the end users by request. The second part is the respective robotic platform which comprises different agents, ways of communication with the RAPP Platform, as well as means of interacting with the low-level aspects of each robot, such as sensors, actuators or even system calls. The following subsection provides a more detailed description of the aforementioned components.

B. Platform side

The RAPP Platform as an entity, can be described as the assorted list of services, processes and data residing within a cloud infrastructure (top part of Fig. 1). Whereas the actual cloud employed may be a single server, a cluster of servers, or a virtual machine on a commercial cloud, from the designer's point of view, we consider this part of the RAPP cloud as the Platform. A distinct component within the RAPP Platform is the RAPP Store, which mainly acts as the web-interface to the rest of the Platform. The robotics application (RApp) is simply an instance of an application provided by a developer, and is distributed through the RAPP Platform after its submission and the appropriate validation process. An actual application may be distributed in nature; however it shall always execute mainly on the robot, and only the distributed parts may have components or utilise services that exist on the cloud. If we assume that a RApp exists in a host/guest fashion, the host can be found in the robot part, which is the main robot controller, whilst the guest may exist in the cloud. The distributed execution offered is a great plus for our implementation, as algorithms that cannot be naturally executed in the entirety of robots can be utilized (i.e. CUDA employment, MPI processing etc.).RApp implementations may range from single robot simple applications to advanced multi-robot cooperation real life paradigms.

The RAPP Improvement Center (RIC) is a RAPP independent module (or complex of modules), which may operate independently (off-line learning), or upon request from a RApp (on-line learning). Thus, it may not be directly part of a RApp, but as a service, it may be accessed through the RAPP API (see Section IV). In a real-world scenario, RIC will be a collection of ML or DM processes, accessed via the API through the aforementioned provided services.

The RAPP Platform architecture further employs an ontology that is used as the knowledge representation of the participating robots. RAPP utilises existing robot ontologies, and extends them in order to allow for the differences required for the specific use cases. The state of the art robot ontology repository that is officially supported by ROS is KnowRob. KnowRob contains four groups of OWL ontologies: Base ontologies, Semantic Robot Description Language, Computable definitions and Semantic environment maps. Additionally KnowRob has a very detailed documentation that is expected to boost the ontology manipulation. Finally, we are also exploring the option of extending KnowRob, as other extensions are already "officially" included. Extensions may be offline (insert ontology elements beforehand for use) or even online if possible (update the knowledge connections during execution). Certain inference and reasoning methodologies will be investigated, in order to allow for potentially high-level robot functionality. Obviously the reasoning mechanisms will exist on the cloud, instead of the robots themselves, offering flexibility in managing computational sources.

One of the architectural strengths of the RAPP Platform, is its ability to serve on-demand requests by robots via HOP services. As defined earlier, HOP is the component that connects each robot to the cloud, and serves as the reconfigurable execution environment for RApps. In specific, HOP is used for communication between the cloud and robot parts of a RApp, as well as for the RApp's requests to various services that are provided by RAPP. These may include general purpose services (e.g., for personalised data acquisition), which bind to ROS nodes, as well as services for updating the robot's knowledge database. The same HOP technology enables to invoke third party web services, even those not specifically designed for robots. At last, HOP facilitates communications with web clients, such as tablets and smartphones, which can be used as user interaction components with the RAPP Platform and some robots, and also for their own computing resources.

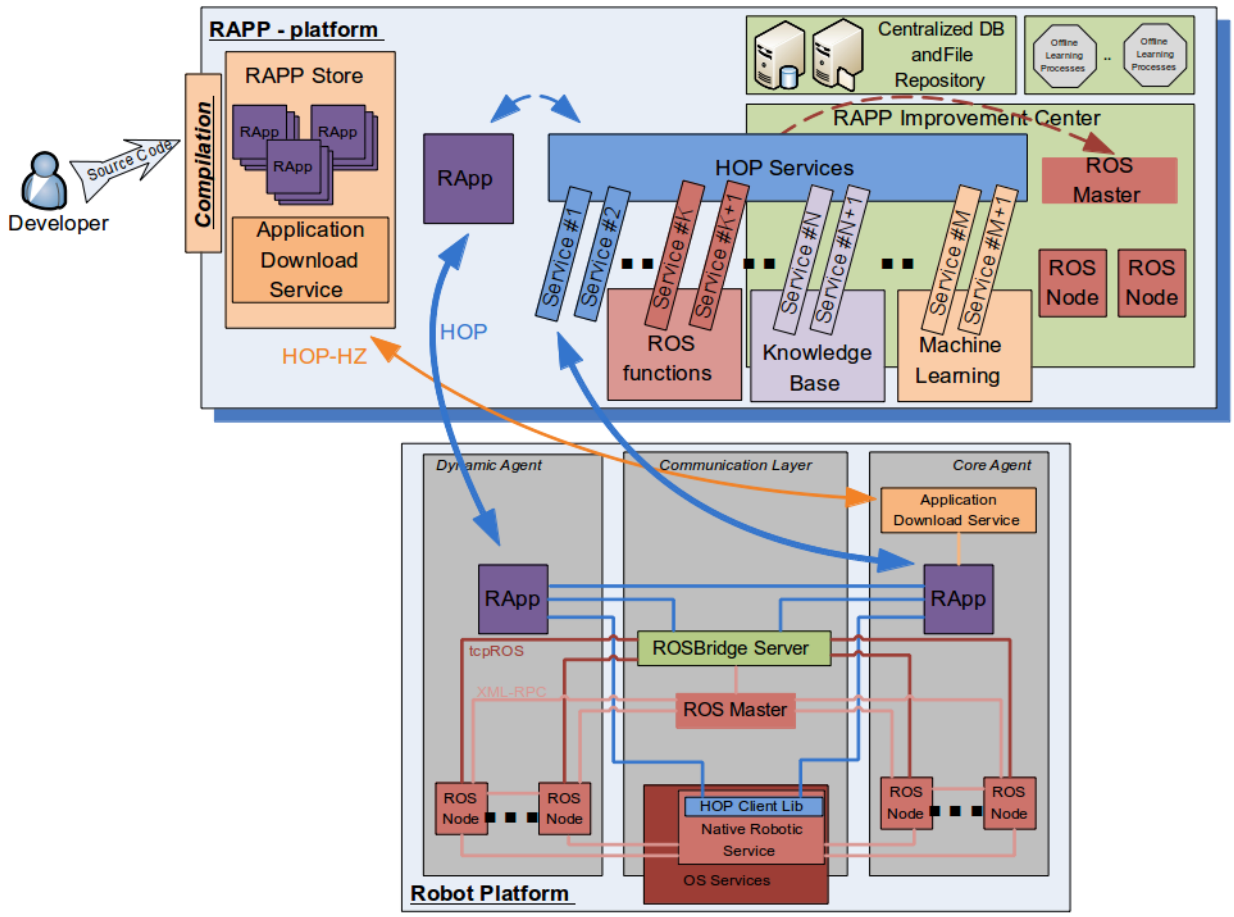

Fig. 1. Overall RAPP system architecture

RApps are programmed using a super-set of JavaScript, which gets compiled and executed by the HOP run-time engines on the robot or the RAPP Platform. As such we are not reinventing the wheel, but we are creating a toolset of libraries and application distribution and execution. HOP provides the logistics for Rapp discovery, downloading from the RAPP platform and installation to the Robot Platform, as well as provides services for authenticating the user, caregivers and devices, and for the personalization of the RAPP and Robot platforms depending on the user profile and robot capabilities. Finally, HOP provides a flexible means to combine ROS services across independent ROS graphs (set of ROS nodes attached to the same Master Node), allowing in particular ROS Nodes on the RAPP platform to be shared by several Robot ROS graphs

A significant design aspect of the RAPP Platform, is the fact that it embeds an instance of ROS master, and may furthermore contain several ROS nodes of heterogeneous functionality. The fact that ROS is an abbreviation of the words ”Robot Operating System” does not necessarily mean that it can only be employed in robotic devices. Since the robotics community extensively supports ROS by providing a vast number of high-quality open-source software packages, ROS can serve multiple functionalities, even on standard PCs. In our case, we use general purpose ROS nodes that uptake the responsibility of executing heavy-duty algorithms on demand, robotic and otherwise, when asked from the respective robots.

Another aspect of the RAPP Platform is depicted on the top-right side of Fig. (1), and is an ensemble of offline learning processes. Admittedly, updating the total collective knowledge obtained by a group of robots, requires procedures that utilise the whole gained dataset. Consequently, and given the expected size of the knowledge database, these procedures are computationally expensive and time-consuming, a fact that sets online learning at a significant disadvantage for most practical purposes. Thus, certain offline procedures exist that will implement the knowledge update at predetermined time intervals.

Finally, a cloud database and file repository are also employed. This storage element addresses the needs of saving all personalised and general purpose information, in order to be consequently available to RIC, RAPP Platform ROS nodes, HOP services, the batch offline procedures, and the to the RApps themselves. Privacy and security of the data contained within in the DB is of paramount importance, as one registered robot must have access only to the information the corresponds to it. The storage DB is segmented per user

per app, thus is not centralized. This fact allows for easily creating personalized robotic applications. A default version of each application will be initially downloaded and via the ontology mechanisms and the machine learning procedures will ”learn” and store in the appropriate place personalized data that correspond to specific users. This way the same RAPPs may operate differently, having binded to different individuals. As far as security is concerned, Virtual Machine Isolation, encypted databases and directories, and all the industry standards with respect to hashing and encrypting sensitive information will be employed.

C. Robot side

The robot-side RAPP architecture (lower part of Fig. 1), consists of several modules that enable both the semi-autonomous robot functionality in its surrounding workspace, and the communication with the RAPP Platform entity, in order to perform heavy duty algorithms and make general knowledge acquisition requests.

A bottom-up description of the the robot-side architecture, starts with the lowest layer, the operating system (OS) which implements the OS and the native robotic services. The term OS services, ascertains the required functionalities of the respective robot's OS, whilst the native robotic services make use of the OS services in order to access sensor measurements and command actuators, via an ensemble of communication protocols. The next level of abstraction contains the ROS meta-operating system and other programs connected to HOP using the HOP C client library. A ROS master instance is executed in the robot OS from the robot's boot up sequence, which combined with the appropriate ROS nodes, provides a higher level of abstraction concerning the native robotic services. HOP C clients are simply executables that do not need ROS to run, but include direct calls to specific native robotic or OS services. It must be stated that the robot-part architecture is quite flexible, since the low level parts of the robot may be accessed by ROS nodes, HOP C clients, or both of them. In specific, and within the context of the RAPP project, two distinct robots are utilised: NAO from Aldebaran [30], and ANG-Med from INRIA [31]. In the RApp implementation, NAO will utilise ROS nodes as provided by ROS community, whereas ANG-Med will employ only HOP C clients specifically created for this robot.

The next robot architectural level describes RApps. A RApp is a collection of HOP source code, ROS nodes wrapped by HOP code, and configuration or data files in general. A RApp is considered to be the controlling part of the robotic agent that operates the robot, and each RApp is capable of having direct communication with the HOP services on the Platform side. This kind of communication may refer to configuration data retrieval, heavy duty algorithms executing, installing, or updating another RApp. The later is achieved with the Application Download Service which is directly connected to the RAPP store via HOP-HZ services, which enables it to download new RApps or new versions of RApps and install them on the robot. The connection of RApps to the ROS nodes is managed through ROSBridge Server, which provides a JSON API to ROS functionality to non-ROS programs.

In contrast to the bottom-up description of the robot architecture, one can perform a vertically layered approach. According to this, the robot consists of three distinct layers: the Core Agent, the Dynamic Agents and the Communication layer. The Core Agent layer (lower right side of Fig. 1), is considered to be pre-installed to robot and contains the Application Download Service, the core RApp and the necessary ROS nodes for the core RApp to operate. The Core Agent's purpose is to provide the basic means interaction with the end user, by which he or she, may dictate the installation, update or removal of RApps (using the Application Download Service), or trigger fundamental robot functions, such as a shut down request. The difference between the Core Agent and the Dynamic Agents, is that the later do not exist a priori in the robot, but their fetching and execution is invoked by the Core Agent, hence the description ”Dynamic”. In other words, the Dynamic Agents are RApps that provide functionalities which are user, robot and time specific, whereas the Core Agents provide the means to manage the Dynamic Agents. In between, the Communication layer exists that enables the direct connection of RApps to the lower level robotic functionalities and composes of the ROS master, ROSBridge and the Robotic and OS specific calls.

IV. RAPP API

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 design of RAPP API is the integrating part of the whole system. It allows for the cross-robot development of binaries, and provides a high-level abstraction of the base functions via a RAPP wrapper.

The API should provide the functionality of ROS, HOP and any of the RAPP developed functions. Such functions include, but are not limited by, machine learning algorithms, statistical procedures, data analysis etc. In future versions of the RAPP API, these functions will be updated to allow for higher complexity modules. The developer should be aware which components will ultimately run on the robot and which on the cloud, but should not generally have to get familiar with lower level development regarding intercommunication, interaction or synchronisation. It's an integral part of the RAPP project that each application should eventually be encoded by the respective developer as a single RApp, which will transparently encompass the distributed aspect of the RAPP Platform.

The benefit of distributed robotic controllers can only be fully achieved, if we enable the creation of distributed applications in an easy and convenient manner, so as to attract developers from across disciplines, similar to how smart-phone development rose through the app paradigm.

This implies that the API will provide a significant level of abstraction to the developers, which may impose certain restrictions. We are willing to create a closed ecosystem for the development of such applications, if and only if, the gain is greater than those imposed restrictions.

As an example, given that a developer can quickly create an application for a robot, using the RAPP's API provided modules for object recognition, voice recognition, motion planning, navigation, and other complicated and computationally intensive tasks, then it may be worth imposing those restrictions on certain programming practices. In practical terms, we remove the lower-level specifics of a robot systems from the developer. Similar to the current approaches to developing multi-threaded applications, one does not want to be burdened with the specifics of low level threads; instead we want a developer using our API to simply use the high level abstracted function and classes.

The RAPP API is in its design phase, so a specific list of functions may be premature and misleading at this point. However, the APIs design will allow both for online services (such as user preferences, person identification, computer vision, object recognition, object tracking or sound/audio/voice recognition) and for computationally expensive offline services (such as data mining and machine learning processes).

V. CONCLUSIONS - FUTURE WORK

This paper describes the architectural design of the RAPP framework, which comprises two parts: the RAPP Platform, which is the web and cloud-part of the RAPP overall architecture, and the robot-side software, which resides in each robotic device. Both parts are specifically designed in order to maximise flexibility, while the core functionalities are always existent. In specific, a robotic developer may create RApps that are purely HOP-based, or he could incorporate ROS nodes that will be executed on the robot. In addition to that, a RApp is flexible enough to be executed in a distributed manner, with partial execution on the cloud and partial execution on the robot. Regarding the robot side, any robot that has a Unix-based OS, in which HOP or ROS can be installed, is supported. Different behaviours (RApps) may be invoked at run time via the Core RApp Agent, and new information can be sent to the RAPP Platform. Finally, RAPP Platform contains the RAPP Store, which handles the packaging and the distribution of RApps, and RIC, which handles the overall robotic knowledge representation and inference, and HOP services that enable the on-request execution of heavy-duty procedures (such as image processing).

Conclusively RAPP platform offers distributed execution, homogeneous development regardless of platform/robot and most important of all a cloud / robot seamless platform.

An example RApp executing on a NAO robot is the *Find Lost Objects* Application. An elder lives on his own and begins to suffer from mild cognitive impairment, fact that prohibits him from living entirely independently. His son gifts him a NAO robot and one of the applications installed is the aforementioned. One day the elder looses his/her keys and asks his companion to find them. NAO contacts the cloud and asks if there is a common area for humans to misplace their keys. Unfortunately, no such information exists and NAO decides to search the house utilizing primitive localization algorithms and image processing cloud services and eventually finds the keys at the hallway. Then it fetches the keys to the elder and in the same time uploads non-personalized information that the "keys" object can be likely found in the "hallway" location, in order for the other robots to benefit from this knowledge. At the same time, it securely updates the specific person's cloud information about his/her tendency in loosing the keys and the place he/she tends to misplace them.

Future work includes the implementation of the RAPP architecture and the release of the RAPP Platform as a publicly accessible framework. It must be stated that, within the project's lifetime, the robot-side architecture will be tested on two completely heterogeneous robotic devices. The first one is NAO from Aldebaran, which is an anthropomorphic robot with 25 degrees of freedom. It is equipped with an Intel Atom 1.6 GHz processor and its OS is OPENNAO, a Gentoo-based Unix distribution. The second robot is ANG-Med, a smart rollator which includes sensors like 3D accelerometers, gyroscopes, automatic brakes and encoders, as well as a PC with a Linux distribution. Due to the nature of our specifications, these two robots, as well as any other robot, with an Unix-based OS can be a part of the RAPP ecosystem.

ACKNOWLEDGMENT

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

REFERENCES

[1] Lutz, Wolfgang, Warren Sanderson, and Sergei Scherbov. "The coming acceleration of global population ageing." Nature 451, no. 7179 (2008): 716-719.

[2] Bemelmans, Roger, Gert Jan Gelderblom, Pieter Jonker, and Luc De Witte. "Socially assistive robots in elderly care: A systematic review into effects and effectiveness." Journal of the American Medical Directors Association 13, no. 2 (2012): 114-120.

[3] Fujita, Masahiro. "AIBO: Toward the era of digital creatures." The International Journal of Robotics Research 20, no. 10 (2001): 781-794.

[4] Kanda, Takayuki, Takayuki Hirano, Daniel Eaton, and Hiroshi Ishiguro. "Interactive robots as social partners and peer tutors for children: A field trial." Human-computer interaction 19, no. 1 (2004): 61-84.

[5] Pollack, Martha E., Laura Brown, Dirk Colbry, Cheryl Orosz, Bart Peintner, Sailesh Ramakrishnan, Sandra Engberg et al. "Pearl: A mobile robotic assistant for the elderly." In AAAI 2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care, pp. 85-92. 2002.

[6] Schulte, Jamieson, Charles Rosenberg, and Sebastian Thrun. "Spontaneous, short-term interaction with mobile robots." In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, vol. 1, pp. 658-663. IEEE, 1999.

[7] Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. "ROS: an open-source Robot Operating System." In ICRA workshop on open source software, vol. 3, no. 3.2. 2009.

[8] Joyeux, Sylvain, and Jan Albiez. "Robot development: from components to systems." In 6th National Conference on Control Architectures of Robots. 2011.

[9] Osentoski, Sarah, Graylin Jay, Christopher Crick, Benjamin Pitzer, Charles DuHadway, and Odest Chadwicke Jenkins. "Robots as web services: Reproducible experimentation and application development using rosjs." In Robotics and Automation (ICRA), 2011 IEEE International Conference on, pp. 6078-6083. IEEE, 2011.

[10] Roalter, Luis, Matthias Kranz, and Andreas Mller. "A middleware for intelligent environments and the internet of things." In Ubiquitous Intelligence and Computing, pp. 267-281. Springer Berlin Heidelberg, 2010.

[11] Elkady, Ayssam, Jovin Joy, and Tarek Sobh. "A plug and play middleware for sensory modules, actuation platforms and task descriptions in robotic manipulation platforms." In Submitted to Proc. 2011 ASME International Design Engineering Technical Conf. and Computers and Information in Engineering Conf. (IDETC / CIE'11). 2011.

[12] Beetz, Michael, Lorenz Mosenlechner, and Moritz Tenorth. "CRAMA Cognitive Robot Abstract Machine for everyday manipulation in human environments." In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, pp. 1012-1017. IEEE, 2010.

[13] Brian Gerkey, Toward a Robot App Store, IJCAI Workshop on Robotics, 2009, Online: http://robotics.cs.brown.edu/events/ijcai09/materials/gerkey_ijcai09_robotics.pdf

[14] Nielsen, Sren Hundevadt, Anders Bgild, Kjeld Jensen, and Keld Kjrhus Bertelsen. "Implementations of frobomind using the robot operating system framework." In NJF Seminar 441 Automation and System Technology in Plant Production, vol. 7, no. 5, pp. 10-14. 2011.

[15] Wang, Wei, Benjamin Johnston, and Mary-Anne Williams. "Social networking for robots to share knowledge, skills and know-how." In Social Robotics, pp. 418-427. Springer Berlin Heidelberg, 2012.

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

[17] Kuffner, James. "Robots with their heads in the clouds." Discovery News(2011).

[18] Wang, Wei, Benjamin Johnston, and Mary-Anne Williams. "Social networking for robots to share knowledge, skills and know-how." In Social Robotics, pp. 418-427. Springer Berlin Heidelberg, 2012.

[19] Waibel, Markus, Michael Beetz, Javier Civera, Raffaello d'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Haussermann et al. "A World Wide Web for Robots." IEEE Robotics & Automation Magazine (2011).

[20] Tenorth, Moritz, Ulrich Klank, Dejan Pangercic, and Michael Beetz. "Web-enabled robots." Robotics & Automation Magazine, IEEE 18, no. 2 (2011): 58-68.

[21] Tenorth, Moritz, and Michael Beetz. "KnowRob: A knowledge processing infrastructure for cognition-enabled robots." The International Journal of Robotics Research 32, no. 5 (2013): 566-590.

[22] Tenorth, Moritz, Daniel Nyga, and Michael Beetz. "Understanding and executing instructions for everyday manipulation tasks from the world wide web." In Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 1486-1491. IEEE, 2010.

[23] Tenorth, Moritz, Ulrich Klank, Dejan Pangercic, and Michael Beetz. "Web-enabled robots." Robotics & Automation Magazine, IEEE 18, no. 2 (2011): 58-68.

[24] Saffiotti, Alessandro, Mathias Broxvall, Marco Gritti, Kevin LeBlanc, Robert Lundh, Jayedur Rashid, BeomSu Seo, and Young-Jo Cho. "The PEIS-ecology project: vision and results." In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pp. 2329-2335. IEEE, 2008.

[25] Ha, Young-Guk, Joo-Chan Sohn, Young-Jo Cho, and Hyunsoo Yoon. "Towards ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework." ETRI journal 27, no. 6 (2005): 666-676.

[26] Sanfeliu, Alberto, Norihiro Hagita, and Alessandro Saffiotti. "Network robot systems." Robotics and Autonomous Systems 56, no. 10 (2008): 793-797.

[27] Serrano, Manuel, Erick Gallesio, and Florian Loitsch. "Hop: a language for programming the web 2. 0." In OOPSLA Companion, pp. 975-985. 2006.

[28] Serrano, Manuel, and Christian Queinnec. "A multi-tier semantics for Hop." Higher-Order and Symbolic Computation 23, no. 4 (2010): 409-431.

[29] M. Serrano and G. Berry. Multitier Programming in Hop - a first step toward programming 21st-century applications. Communications of the ACM, 55(8):5359, Aug. 2012

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

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