Proceedings of 2017 4th International Conference on
Control, Decision and Information Technologies (CoDIT'17) / April 5-7, 2017, Barcelona, Spain

A Web services based solution for the NAO Robot in Cloud Robotics environment

Radhia Bouziane
University of Biskra,
LINFI Laboratory
Department of Computer Science
P.O.Box 145 RP, 07000,
Biskra, ALGERIA
Email: radia.bzne@gmail.com

Labib Sadek Terrissa
University of Biskra,
LINFI Laboratory
Email: terrissalabib@gmail.com

Soheyb Ayad
LINFI Laboratory
Email: ayad_soheyb@yahoo.fr

Jean-Franois Brethe
GREAH Laboratory
University of LeHavre, France
jean-francois.brethe@univ-lehavre.fr

Okba Kazar
LINFI Laboratory
Email: kazarokba@yahoo.fr

***Abstract*—Since a few years, the development of robots introduced the Cloud Computing paradigm in robotics field as "Cloud Robotics" generation. This new concept allows robots to outsource computing capabilities over the Cloud, whither the Computing involves extra power requirements that could reduce the duration of the process. In this paper, we outline a new Cloud Robotics architecture for NAO robots. Based on Robot Operating System (ROS) middleware, this approach integrates Web services technologies in a Cloud Computing environment, in order to enable Nao robots to consume their packages as an on-demand solution (as a service).**

***Index Terms*—Cloud Computing, Cloud Robotics, NAO, ROS, Web services.**

I. Introduction

For several years, the development of robotics shows the growing interest of introducing robots in different fields like manufacturing technology, medicine, agriculture, underwater investigations...etc. Currently, humanoid robots have an important situation in the research area, and seems to be integrated in our daily life, including ASIMO robot, REEM-C, Pepper, the NAO robot (see Figure 1) and so on. However, a major challenge of this area is to produce a kind of robots that can perform complex tasks in their environments autonomously, and to improve their interaction with humans. In 2010, the concept "Cloud Robotics" was born to benefit from the powerful computing resources of the Cloud [1]. The National Institute of Standards and Technology (NIST) defines Cloud Computing as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [2].

The development of Cloud services comprises three different types: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [2]. This new style provides a major change in applications development. In place of running programs and storing data in a local computer, all the work will be hosted in the Cloud [3]. By introducing these concepts in robotics field, robots can offload complex tasks like image processing, navigation, object or voice recognition, and access all knowledges and software on 'pay as you go' mode.

Fig. 1. The NAO robot.

Cloud Computing paradigm imports many new technologies and architectural designs to Service-oriented architecture (SOA). SOA uses the concept of loosely coupled integration to create, organize, and reuse services [3]. Indeed, using SOA with Cloud Robotics improves the availability and scalability of robotic services, in the same manner of using Cloud Computing with SOA.

Web services technology knew a large acceptance for implementing SOA [4], which enables the reuse of services due to their independence of any programming language and platforms, by following the industry standards such as Web Service Definition Language (WSDL) and Universal Description Discovery and Integration (UDDI). The communication between the services can be managed through REST or SOAP Protocol.

The aim of this paper is to present a new Cloud Robotics architecture that exploits the mechanism of SOA for NAO robots, by using SOAP Web services. In our design, the NAO plays the role of a Cloud Customer that benefits from the computational resources, and the necessary software to perform and share its required task. This work is based on Robot Operating System (ROS) as a robotic middleware.

The remainder of our paper is structured as follows. Section 2 introduces the Aldebaran NAO robot and the Robot Operating System. In section 3 we encompass the related works around our topic. Section 4 presents the proposed system architecture. Section 5 outlines our technical requirements and the implemented solution. We conclude this paper by Section 6.

II. NAO and Robot Operating System

The ROS system is compatible with many robots like TurtleBot, PR2, and NAO robot. In this section, we describe the NAO robot and ROS, which are used in this work.

A. The humanoid robot NAO

NAO is the first humanoid robot that was developed by the french manufacturer of humanoid robots “Aldebaran-Robotics”. With its height of 58cm and its improved shape (see Figure 1), NAO is now used in 70 countries around the world as a research and education platform [5]. It is equipped with 4 microphones, 2 HD cameras, 9 touch sensors, 8 pressure sensors, a 1,6 GHz CPU, 8 GB for storage and 25 degrees of freedom [6]. Regarding to robots functionalities, NAO was designed to have 7 senses for natural interaction: Moving, Feeling, Hearing, Speaking, Seeing, Connecting to Internet autonomously, and Thinking to reproduce human behaviour [5].

B. Robot Operating System

Robot Operating System (ROS), is an open-source metaoperating system for robots that provides hardware abstraction, low level device control, package management etc [7]. It has become as one of the widely used robotics systems these days. Its philosophy is to support software reuse in robotics development by making a little change in codes for each robot. ROS provides several client libraries such as roscpp, rospy, roslisp, and rosjava that support the implementation of ROS programs with different programming languages. The computation network in ROS called the computation graph. The main concepts of this level are [7]:

- Nodes: In ROS, each process that perform computation called a node. Nodes are written with the use of a ROS client library.
- Messages: Nodes communicate with each other via messages that represent data structures (integer, String...).
- Topics: Each message in ROS is transported using Topics. A node can publish a message on a topic, while other nodes can subscribe to read the written data.
- Services: In some cases (when we work with a distributed system), nodes need a request/response mode for communication via services.
- Master: The ROS Master provides name registration and lookup to the rest of the nodes. It manages all nodes together.

In the Figure 2, we can see the graphical representation of this level for NAO robot (using the command rqt_graph: rectangles containing a circle define nodes, and single rectangles are topics).

III. Related Works

In this section, we present some existing works by classifying them in three subsections that deal with the major keys of our research axes:

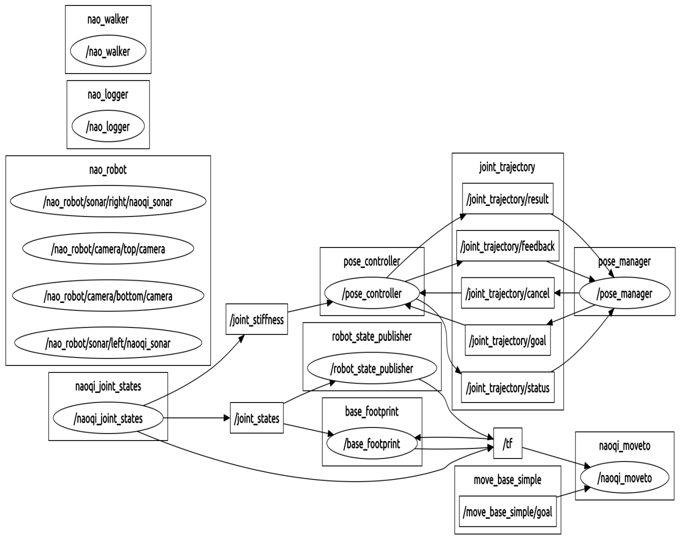


Fig. 2. The ROS computation graph of NAO.

A. Cloud Robotics

In the past few years, many projects have been produced around Cloud Robotics [12][13]. The paper [8] presents the emergence of this concept, and some of its projects such as RoboEarth [9], Cloud-based robot grasping [10], Rapyuta [11], and MyRobots project. Guoqiang Hu, et.al describe the challenges in networked robotics, and propose a Cloud Robotics architecture in [14]. The architecture leverages the combination of machine-to-machine (M2M) level, where a group of robots communicate via wireless links, and machine-to-cloud (M2C) level, that allows robots to offload computation-intensive tasks in Cloud infrastructure. In [15], the authors propose a new Cloud Robotics architecture by offering a new layer, which is the Virtual robot Layer compared with Cloud Computing architecture. The proposed layer includes two essential components: Robot Management System and the Virtual Robot System.

B. SOA and Web services technologies in Cloud Robotics

We give here a brief overview of some works providing Service-Oriented solutions to robots using Cloud concepts:

1) SOA in Robotics: Service Oriented Architecture (SOA) has been applied to robotics in many studies [16]. The paper [17] presents a systematic review of the development of Service-Oriented Robotic Systems (SORS). The authors of this work noticed the increment of SORS that use in particular existing technologies, which is shown through the large use of Web service standards. In [18], almost the same authors present a proposed tool (RoboSeT) that supports cataloging, and discovery of services for SORS. This mechanism is based on semantic information provided by a taxonomy of such services.

2) Introducing SOA and Web srvices with Cloud concepts for robotic solutions: Yinong Chen, et.al proposed the concept of Robot as a Service (RaaS) in [19] as Cloud Computing

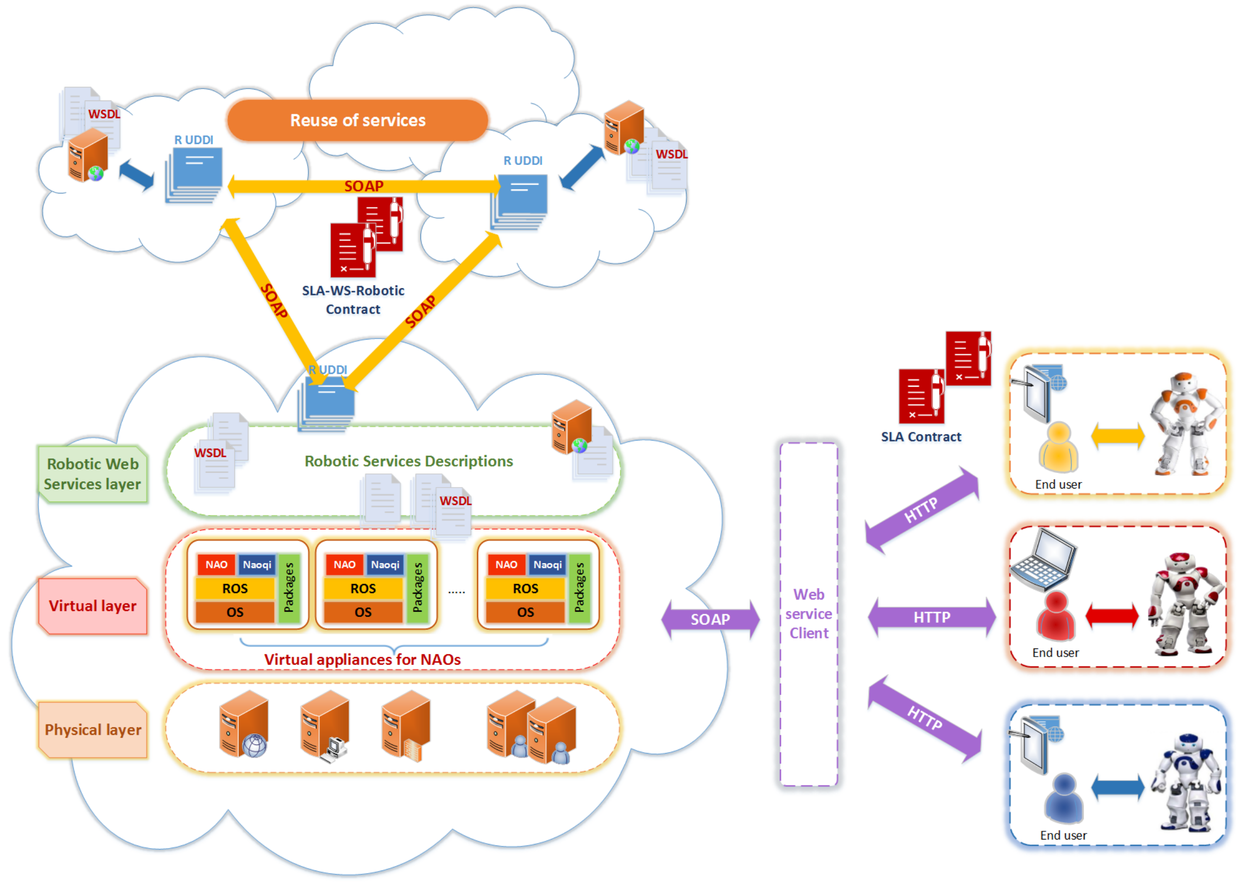


Fig. 3. Main Cloud Robotic architecture for NAOs.

unit using SOA concepts. Users obtain access to use robotic services, and applications from a powerful remote processor, using Microsoft Robotics Developer Studio (MRDS). The authors tested their solution on different Intel processors to express the performance of RaaS. SOA and MRDS are also used in [20] that presents an approach of robotic services in Cloud Computing. The authors introduce, in the proposed system, standard protocols to interact with robotic services such as HTTP and SOAP. In [21], A. Koubaa proposed the RoboWeb system that allows different students and researchers to access and use robots through the Internet (remote robotic lab). The objective of this work is to introduce the SOA approach in order to virtualize robotic resources. The solution is based on SOAP Web services that interact with Robot Operating System (ROS) middleware of robots. In the other hand, the paper [22] provides a new level of abstractions for ROS. This time, A. Koubaa integrates both SOAP and REST Web services into ROS, by merging ROSJAVA library with JAX-WS and JAX-RS Web services.

C. *ROS on Web*

In [23], the authors presented the proposed Rosbridge technology that exposes ROS functionalities to developers as a simple interaction over sockets. The Javascript library "rosjs" has been developed on top of rosbridge, which allows to create Web applications for robots. The paper [24] presents a Web-based Remote Robotic Laboratory for PR2 robots, using the proposed Rosbridge technology [23] that exposes ROS functionalities to developers as a simple interaction. Through the paper [22], independent client applications can use ROS programs by invoking the ROS Web services in the same manner as invoking traditional Web services.

Indeed, we notice that there is a lack of using classical Cloud Computing architecture in Service-Oriented based works including: Computing resources, IaaS and virtual machines (Even in works that uses this paradigm as a solution). For this reason, we extend the system architecture of [15] with Web services technologies where the Client robot NAO can consume computing capabilities (hardware part and software components) provided by multiple providers, in order to perform a required task. We consider SOAP Web services as connections tools, to ensure the communication between the Clouds services.

IV. Proposed Architecture

Cloud Robotics is a way of delivering robotics capabilities to users in the form of "services" according to SLA contracts. In this context, we have proposed a Cloud architecture for

NAO robots which is shown in the Figure 3. In our design, the NAO plays the role of a Cloud Customer that benefits from the computational resources, storage, and the necessary software to perform its required task. Users can interact with their virtual robotic solutions, through simple Web interface to handle and use their robots. There are three levels in the Cloud side:

1) The physical level: it represents the set of physical resources, such as storage servers, network equipments...etc.
2) The virtual level: it contains virtual machines, which enables to provide multiple applications on multiple instances to serve a large number of customers.
3) The Robotic Web services level: in this layer, the robotic software are exposed as Web services.

The description of this architecture and its functioning will be presented in the following subsections.

A. Goals of system architecture

Our approach is designed to provide two main requirements:

1) Using virtualization to offer robotic hardware to NAO robots: Unlike traditional robots that use a large number of devices for storage, calculation and processing, they can now use new types of hardware, less expensive, but more effective. Users can access to their virtual robotic appliances hosted on Cloud servers over Internet. These appliances represent the technical platforms and tools to run NAO based programs. They are composed by (as shown in Figure 3):

- An operating system (Ubuntu).
- The middleware ROS.
- NAO robot stacks.
- The NaoQi SDK [25] to control NAO.

The Cloud provider offers this virtual layer to customers on pay-as-you-go mode to define the usage (duration of use, quantity and quality of resources), and take the responsibility of installation, control, and maintenance...etc. The relationship between the service provider and consumers is regulated by a service contract (Service Level Agreement -SLA [3]).

2) Benefit of robotic software for a specific task: On the software side also, many robotics software (like navigation algorithms, object or voice recognition solutions) are delivered to be consumed as a service. Users through internet, and from any device like mobile phones or laptops, can consume these services to help their connected Client robots to accomplish an assigned task. Indeed, These on-demand services are delivered and distributed to customers by various Cloud services providers over Internet. One of the major keys is to make these services available on different Cloud platforms over the internet, and to ensure the communication between Clouds. In this case, SOAP Web services can be established as a connection mechanism, which implements SOA architecture, to assure the exchange of requests and responses among services, based on SOAP protocol. By exposing the software as Web services for robots, we can benefit from all the advantages of traditional Web services such as:

TABLE I
TECHNICAL TOOLS

Tools	Requirements
Robot	NAO
Operating system	Ubuntu 12.04 LTS
Robotics system	ROS (Hydro version)
ROS Library	rosjava
NAOqi SDK	naoqi-sdk-2.1.2.17-linux32
HTTP Image streaming server	web_video_server
Remote shell programming	sshxcute-1.0.jar
Development environment	eclipse (jee-indigo version)
JDK and APIs	openjdk-7-jdk, JDBC (mysql-connector)
Web technology	JSP
Web server	apache-tomcat-7.0.59
Database Server	mysql (Server version: 5.5.43)
Apache Web services engine	Apache Axis2 (version: 1.5.1)

- Interoperability and exchange of messages between different applications.
- Reuse services over Clouds.
- Data protection: Web services allow the exchange of messages between different applications without knowing the background on appropriate data or codes.

B. System functioning

The execution scenario of our approach can be summarised in these two main process:

1) Publication of services, and Establishing contracts between Cloud providers:
    a) Publication: Each local service provider publishes its Web service description in the local Robotic Universal Description, Discovery and Integration (R UDDI) registry.
    b) Establishing contracts between Cloud providers and synchronization between repositories: Our approach represents a multi Clouds system that is composed by multiple providers. Every service description can be registered in an external registry R UDDI of another Cloud in order to be used over Clouds (see Figure 3). These collaborations has been determined in services contracts (SLA-WS-Robotic contracts). The Cloud contracts identify the consumers and the Cloud services provider, and describe the robotic Web services features.
2) Consumption and Invocation of robotic Web services:
    a) Research: Searching for desired services that matche consumers criteria is established by a Web server. If the service exists, the registry provides the consumer with the endpoint address for the service.

b) Invocation: The service consumer invokes the desired Web service to retrieve a result of treatment.
c) Invocation and Execution in ROS: The application invokes a Web service and retrieves desired results that can be used in ROS application to perform robotic task.
d) Actions in NAO: After running the ROS application, we can see the robot performs the required task.

V. Technical Requirements and Implemented Solution

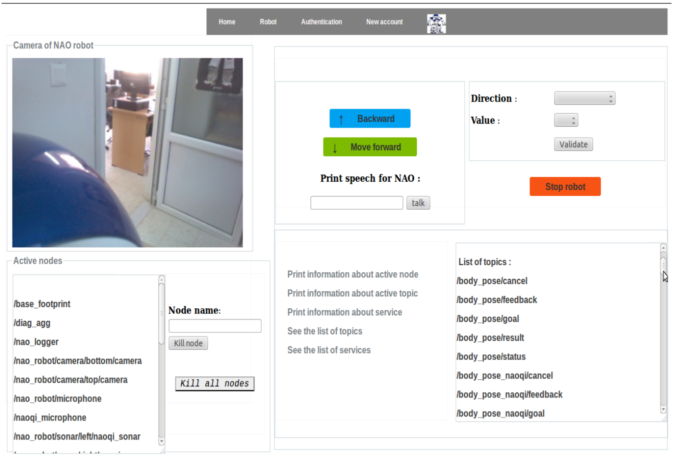
In order to test the fuctionnig of the architecture on our real NOA robot (in LINFI Laboratory), we used the tools mentioned in Table I.

- We used the "Hydro" version of ROS as robotic middleware which is turning on Ubuntu 12.04 LTS system.
- To use NAO with ROS, we need to install the Aldebaran NAOqi SDK [25].
- rosjava is a ROS based library, which allows to program ROS in java. We used it to create our personal robotic packages. We used this library in Eclipse environment (under openjdk-7). Eclipse offers us many advantages due to its modular design, based on a plugin loading motor that makes Eclipse a upgradeable toolbox. Eclipse allows us also to develop our Robotics Web solutions, as any Web applications using JSP.
- Using rosjava with Eclipse [26][27], and sshxcute for ROS Remote shell programming allow us to run and to display ROS robotic tasks.
- To realize the customers Web interface (see Figure 4) which allows the robot control , we used the technology JSP (Java Server Page).
- For video streaming, the web_video_server [28] has been used.
- We used the core engine for Web services "Axis2" to develop SOAP Web services.
- mysql server has been used as database server. We need it to register information about users and NAO. In order to access to the database from java, we used the JDBC API.

The Figure 4 shows the features of customers Web interface. It allows users to control the NAO by executing and displaying the following operations:

- start the NAO.
- move the robot in different directions.
- stop it.
- make NOA talk a speech.
- see the list of nodes, topics and services of NAO.
- get information about a desired node, topic or service.
- kill all nodes, or a particular one.

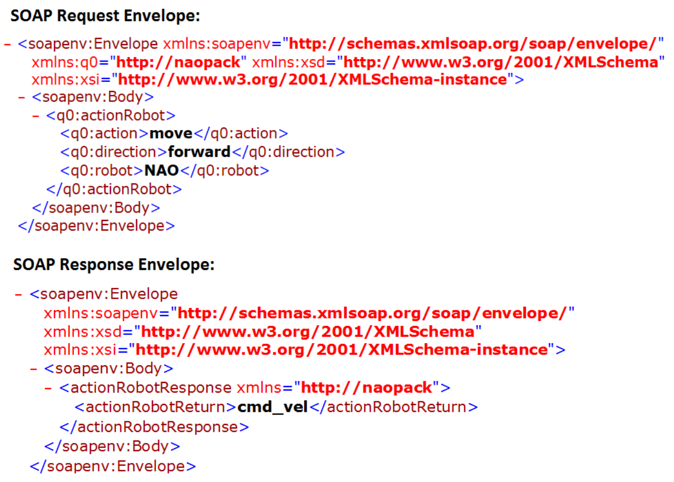
To accomplish These operations, the robotic application uses SOAP Web services to get specific information about the robot (like topics, nodes...) in order to run robotic tasks. These



/base_footprint
/diag_agg
/nao_logger
/nao_robot/camera/bottom/camera
/nao_robot/camera/top/camera
/nao_robot/microphone
/naoqi_microphone
/nao_robot/sonar/left/naoqi_sonar

List of topics :
/body_pose/cancel
/body_pose/feedback
/body_pose/goal
/body_pose/result
/body_pose/status
/body_pose_naoqi/cancel
/body_pose_naoqi/feedback
/body_pose_naoqi/goal

Fig. 4. Client Web interface.



```
SOAP Request Envelope:

- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:q0="http://naopack" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soapenv:Body>
    - <q0:actionRobot>
        <q0:action>move</q0:action>
        <q0:direction>forward</q0:direction>
        <q0:robot>NAO</q0:robot>
      </q0:actionRobot>
    </soapenv:Body>
  </soapenv:Envelope>

SOAP Response Envelope:

- <soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <soapenv:Body>
    - <actionRobotResponse xmlns="http://naopack">
        <actionRobotReturn>cmd_vel</actionRobotReturn>
      </actionRobotResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Fig. 5. SOAP Request and Response Envelopes.

process are exposed as SOAP Web services. Figure 5 shows the Request and the Response Envelopes in case of invoking the Web service of NAO movements. The envelopes indicate the communication between the Web service consumer and provider through messages exchange. Once the user invokes the Web service, the appropriate topic "cmd_vel" will be obtained as a result. In this case, we will see the process execution, and the robot will perform the required task by publishing ROS messages in the appropriate topic.

VI. Conclusion and Future Works

In this paper, we have introduced a new design for our NAO robot in Cloud Computing architecture. It is an approach based on Web services technologies and ROS. We added a new Cloud communication layer between the Client (robot side) and the Cloud providers, with the necessary components that make the system works for any real scenarios. Our future step will focus on the implementation of the whole system with its different levels, and the validation on real Cloud infrastructure.

References

[1] J. Kuffner, *Cloud-Enabled Robots*. In IEEE-RAS International Conference on Humanoid Robots, 2010.

[2] P. Mell and T. Grance, *The NIST definition of cloud computing*. 2011.

[3] M. Zaigham and R. Hill, *Cloud Computing for enterprise architectures*. Springer Science and Business Media, 2011.

[4] M. Endrei, et al., *Patterns: Service-Oriented Architecture and Web Services*. International Technical Support Organization, April 2004.

[5] *Aldebaran-Robotics company*, www.aldebaran-robotics.com

[6] *Robot humanode NAO*, http://www.erm-automatismes.com/d00013E-robot-humanoide-nao.pdf

[7] *ROS*, http://wiki.ros.org

[8] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, *A survey of research on cloud robotics and automation*. Automation Science and Engineering, IEEE Transactions on, 2015, vol. 12, no 2, p. 398-409.

[9] *What is RoboEarth?*, http://www.roboearth.org

[10] B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner, and K. Goldberg, *Cloud-Based Robot Grasping with the Google Object Recognition Engine*. In International Conference on Robotics and Automation (ICRA), 2013, 4263-4270.

[11] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, *Rapyuta: The roboearth cloud engine*. In International Conference on Robotics and Automation (ICRA). IEEE, 2013, 438-444.

[12] E. Guizzo, *Robots with their heads in the clouds*. IEEE Spectrum 48 (3) (2011) 1618.

[13] A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, and M. Barreto, *Ubiquitous robotics: Recent challenges and future trends*. Robotics and Autonomous Systems, 2013,61(11), 1162-1172.

[14] G. Hu, W.P. Tay, and Y. Wen, *Cloud Robotics: Architectire, Challenges and Applications*. IEEE Network, 2012, 26(3), 21-28.

[15] L.S. Terrissa and S. Ayad, *Towards a new cloud robotics approach*. In Mechatronics and its Applications (ISMA), 2015 10th International Symposium on, pages 1-5. IEEE, 2015.

[16] A. Ahmad and M.A. Babar, *Software architectures for robotic systems: A systematic mapping study*. Journal of Systems and Software (2016), 122, 16-39.

[17] L.B.R. Oliveira, F.S. Osrio, and E.Y. Nakagawa, *An Investigation into the Development of Service-Oriented Robotic Systems*. Proceedings of the 28th Annual ACM Symposium on Applied Computing. ACM, 2013, p. 223-228.

[18] L.B.R. Oliveira, F.A. Amaral, D.B. Martins, F. Oquendo, and E.Y. Nakagawa, *RoboSeT : A Tool to Support Cataloging and Discovery of Services for Service-Oriented Robotic Systems*. Robotics. Springer Berlin Heidelberg, 2015, p. 114-132.

[19] Y. Chen, Z. Du, and M. Garcia-Acosta, *Robot as a Service in Cloud Computing*. In Proceedings of the 5th IEEE International Symposium on Service Oriented System Engineering (SOSE), Nanjing, June 4-5, 2010, pp.151-158.

[20] R. Doriya, P. Chakraborty, and G.C. Nandi, *Robotic Services in Cloud Computing Paradigm*. In : Cloud and Services Computing (ISCOS), 2012 International Symposium on. IEEE, 2012. p. 80-83.

[21] A. Koubaa, *A Service-Oriented Architecture for Virtualizing Robots in Robot-as-a-Service Clouds*. Architecture of Computing SystemsARCS 2014. Springer International Publishing, 2014, p. 196-208.

[22] A. Koubaa, *ROS As a Service: Web Services for Robot Operating System*. Journal of Software Engineering for Robotics. 2015, vol. 6, no 1, p. 1-14.

[23] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, *Rosbridge: Ros for non-ros users*. In Proceedings of the 15th International Symposium on Robotics Research. 2011.

[24] S. Osentoski, B. Pitzer, C. Crick, J. Graylin, S. Dong, D. Grollman, H.B. Suay, and O.C. Jenkins, *Remote robotic laboratories for learning from demonstration*. International Journal of Social Robotics. 2012, vol. 4, no 4, p. 449-461.

[25] *nao*, http://wiki.ros.org/nao

[26] *Using rosjava with Eclipse*, http://wiki.ros.org/rosjava/Build/Eclipse

[27] F. Ellouze, A. Koubaa, and H.Youssef, *ROSWeb Services: A Tutorial*. n : Robot Operating System (ROS). Springer International Publishing, 2016. p. 463-490.

[28] *web_video_server*, http://wiki.ros.org/web_video_server