ASTRO: Architecture of Services Toward Robotic Objects

Jacques Saraydaryan[1], Fabrice Jumel[2] and Adrien Guenard[3]

[1] CPE Lyon
Villeurbanne, 69616, France

[2] CPE Lyon
Villeurbanne, 69616, France

[3] CPE Lyon
Villeurbanne, 69616, France

Abstract

The usage of robots as smart objects becomes a new challenge for research and industrial teams. In this paper, we introduce an architecture of services using and integrating robotic objects. On one hand this architecture allows integrating robotic capabilities into more complex scenarios using other sensors/actuators. On the other hand, robot objects could use our architecture to solve remote problems such as object recognition, map sharing. Heterogeneous services and capabilities could be dynamically used and integrated. The architecture has already been implemented. Global scenarios of usage have been developed illustrating our approach.

Keywords: *Robotic framework, Ontology, Probe-actuator communication homogenization, Global scenario building, Robot Communication Web compliant.*

1. Introduction

The usage of smart objects into large set of use cases such as health-care, smart home or adaptive work environment grows with a continuous need of smarter objects. Robots appear to be a perfect tool for such use cases but the integration with other services, sensors and actuators has suffered from the robot programing and interface complexity.

Currently most of robotic objects bring a high level of usage and programing API. This evolution allows the usage of high level capabilities such as ”go to position”, ”speak” or ”take the relevant object”. ROS (Robotic Operating System) [1] is one of most used robotic middleware but it only covers robotic object and does not target web services. Moreover it provides efficient connections between robot sensors but these connections establishment are static. Naoqi [2] makes great effort to make robotic easily programmable (robotic service proxy, and shared memory for all parameters and service of the robot), however only one type of robot is targeted, remote collaboration and communication are limited.

New challenges appear using such capabilities for complex use cases. One of the key challenge of use cases or services composition achievement is the re-usability as mentioned in [3]. Without such target, new use case or scenario building would become very complex and time consuming. More over the user interaction with service through compliant web protocol for example is limited and the security aspect not completely covered.

Despite lots of related works cover by the Service Object Architecture for various domain (domotic, web service) [4] or internal robotic architecture [5] , few of them address the usage of robotic object as smart objects or offer remote problem solving [3].

In this paper, we intend to provide an architecture of services focused on robotic object integration called Architecture of Services Toward Robotic Objects (ASTRO). This architecture tries to answer to the the following questions; can we integrate robotic object into complex scenario? Can we deliver remote problem solving for robotic object? Is it possible to use dynamically robot capabilities and services? Can we easily create and compose service including robotic objects with high level of capability? Is it possible for the end user to interact with services?

Nowadays more and more robot objects are brought with their own hardware constraints and programming language. Integrating such objects with other controllable objects [6] (electric lamp, sensors, actuators) could allow developers to include a new interaction dimension into their scenario. Some problems need lot of computation resource to be solved (e.g object recognition) and could not be done locally by robots [3]. In order to extend robotic capabilities, some efforts have been made to create a world wide web for robotics [3]. The need of both intelligent service robotics and usage of robot solution capabilities is, as we currently know, not completely covered. Moreover, in heterogeneous and pervasive

environment the usage of service should be dynamically addressed [7]. Calling directly a known service leads to a no scalable solution. No object would know the result of thousand services without services names and objective directory. Because resulted services manipulate robots and actuators, the security aspect must be taken into account in order to avoid services malicious usage and to guaranty body integrity of people. Finally, the design of a high level scenario must not be reserved to experimented developers, the service usage and composition should be easy to use.

In this paper we present an architecture that intent to provide the following points that are not currently covered:

- Enhance user interaction
- Give access to robot and associated service to web service
- Give access to authorize entity from the web to robot and associated services
- Provide an abstraction of robot capacities for service composition
- Integrate security constraint on the architecture

The ASTRO architecture is composed of 5 main blocs: the View bloc, the Relay bloc, the Processing bloc, the Provider bloc and the Resource bloc respectively in charge of the web usage of delivered services (javascript library), relaying services to the Internet through a REST architecture, providing the composition of services (OSGI like), dynamic services and Resources broker and integrating heterogeneous sensors/actuators and robots to our architecture.

This paper is organized as followed, the section II details related work on the subject, the section III provides the detailed description of our architecture, the section IV shows our experimentations and results and finally section V explains our future works.

2. Related Work

As presented in the previous section, we intent to create an architecture able to compose complex scenarios by using services and robot capabilities. This section covers the related works on architecture that provide manipulation capabilities, service composition, service for robot and then architecture dealing with simple capabilities (domotic scenario).

Some significant works have been accomplished for designing and implementing robotic frameworks. Aiming at dynamically wiring different probes and actuators of robot, these architectures allow easy integrating of new capabilities to a robot. [5] provides a survey on such architectures. Authors argue that robotics middleware provides significant advantages such as software modularity, hardware abstraction, platform independence and portability. The different architectures are compared across the following criteria:

- System model, representing the internal middleware engine. e.g from Multi-process architecture [8] [9] to Service-oriented, component-based software [10] [11],
- Control model, defining how reaction and behavior are triggered e.g even-driven [12] [13], message oriented [1],
- Behavior coordination, defining the ability of middleware to coordinate tasks of robots parts or services,
- Dynamic wiring, allowing dynamic configuration of connections between services of components [14] [10],
- Fault tolerance, Supported simulation , Open source nature, Real Time, Distributed Environment and security supported features

Robotic middleware offers key features to interconnect capabilities and highlight the importance of an hardware abstraction and unified access. Some of robotic architectures focus their work on the dynamic wiring aspect [10] simplifying the access to the robot capacities. Moreover [11] design a framework where each basic function is considered as a component with internal activity, input/output connection and command support. In this approach special care are also given to deployment of new component easily [15].

SOM	1	2	3	4	5	6	7	8	9
Cumulus	x	x	x	x					x
Colombo	x	x	x					x	x
SO-CAM		x	x			x			
ubiSOAP		x		x	x		x		
(SI)2	x	x	x	x					
RESCUE		x	x	x	x	x			
OMG DDS		x	x	x		x	x		x
SOA-MM		x	x	x					
QoS SOM		x	x						x
NINOS		x	x	x			x	x	
B-VIS		x	x	x					
SAI	x	x	x		x		x		
Weka4WS		x	x	x	x		x		
CROWN-C		x	x				x		

Fig. 1 Comparison of SOM solution as presented in [4]

Research community (OMG Object Management Group) made also great effort for providing standardization concerning robot service specification and

usage with a first standard specification of Robotic Interaction Service (RoIS) [16]. Current robotic framework and middleware are mainly focuses on delivery services on well known resources. Despite some interest given to Web Compliant communication [?] and service securing, given computation access of robot to the Web Services and easily integrate user interaction are not well covered.

[4] defines objectives of service oriented computing defined by [17] as ”making service available and easily accessible through standardized models and protocols without having to worry about the underlying infrastructures, development models or implementation details”. Service Oriented Middleware (SOM) becomes very important to use service oriented computing in order solve part of SOC (Service-Oriented Computing) issues (supporting heterogeneous environments and systems and providing functional and non-functional requirements for the applications). In [4] and [19] the following criteria are used to compare SOM solutions:

1) Providing a standardized model for service provider

2) Ensuring runtime deployment services mechanism and advertise them (service availability and reliability)

3) Discovering and effectively using published services mechanism for service consumer

4) Hiding the heterogeneity of underlying environments.

5) Making transparent the Service integration to client approach

6) Using an adaptive and autonomous service discovery

7) Scaling up service consumption (access rate, large volume of data, communication bandwidth)

8) Ensuring reliable and secure operation

9) Taking into account QoS

The reviewed SOM solution and associated covered requirements are displayed in figure 1.

We can figure out the (SI)2 solution [18] which provide support for sensors, RFID and embedded device. 2 layers are presented in this approach, one platform dependent layer in charge of handling messages for the smart component and a service lifecycle manager which is responsible for deploying starting and stopping service on component. The platform independent layer gets the service description from the middleware and search for possible deployment. Moreover a request processor module handles the requests for specific service and transmits if to the correct message handler in platform dependent layer. The SOA-MM [19] provides an integration of shop floor equipment (embedded device, RFID, sensor) to business application. The authors suggest an integration layer managing the communication between the actual capabilities and data flows and the services requested by the business application. Service registration, device positioning, information retrieval and event notification process. Both (SI)2 and SOA-MM approach aims at composing services using sensors or embedded systems. The service composition itself is not targeted and the resource qualification is not done dynamically. The main target of these solutions is to quickly and automatically provide devices information to services. the DoMAIns approach [6] is quite different in that they intend to fully qualify the environment and the capabilities. It is not a SOM approach but more an ambient intelligence modeling. In order to well understand interactions between environment and capabilities, 5 different representation domains are defined: User, Environment, Actionable (object that can be ”moved”, e.g. a door), controllable (”smart” object that software can control e.g sensor) and AmI (i.e Ambient Intelligence). Each domain is described through an ontology (DogOnt [20]) that allows to dynamic request actions on appropriate controllable or actionable object and develop automation scenarios. Although DoMAIns do not target the service composition or service representation, the automation scenario creation is possible using ontology reasoner.

The RoboEarth framework [3] aims at helping robot to accomplish complex task (serve a drink to patient into hospital, object recognition). The architecture is composed of 3 layers, robotic specific layer, generic component layer and server layer. The robot specific layer provides a skill abstraction feature allowing communicating with robot in heterogeneous way. The generic component layer uses an action execution module to ensure reliable action execution on the robot. In case of unresolvable problems, user action is needed and asked. Moreover, the layer is responsive of the environment modeling. The environment modeling maintains the global world model state. A semantic map is built that represents the environment map where robot would update detected objects or obstacles. The feedback of robot’s performance task can be evaluated by user through a learning component. This component could adjust robot behavior and strategy regarding to the given feedback. New behavior could be learnt with the Action and Situation Recognition and Labeling component that provide to users the ability to record behavior of tele-operated robot. The server layer stores all necessary data and available services into a database component (storing CAD models, point clouds, image data for objects). [3] is focused on helping robot to resolve complex tasks by providing a robot’s services. Despite they not center their work on the service composition, [3] provides key features that enable to store and use robot capacities and model robot’s world environment.

Related work currently not completely covers our purpose of building complex scenarios using robot’s capabilities and providing services to robot. Architectures

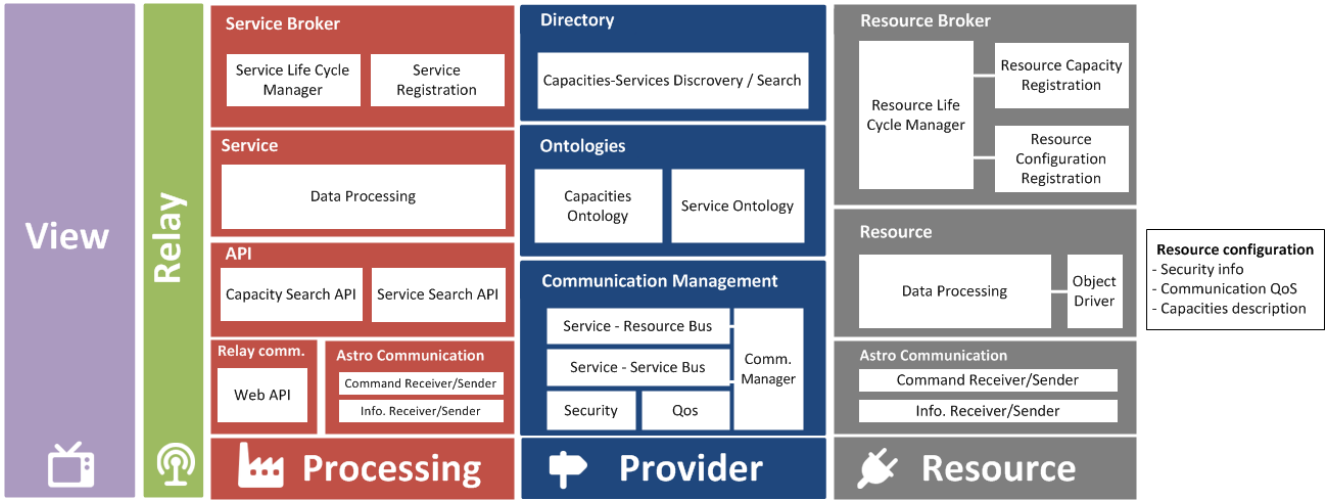

Fig.3 ASTRO Architecture

dealing with robot's capabilities are well covered by the literature as well as the service composition into service oriented middleware. Some architectures [6] initiate the usage of sensors and embedded system but do not provide real middleware of service composition. Recently projects deliver key features for services to robot around the notion of World Wide Web for robots but do not target the complex scenarios creation using robot's capabilities.

In order to cover the complex scenarios building using robot's capabilities issue, we focus our work on the dynamic service composition aspect.

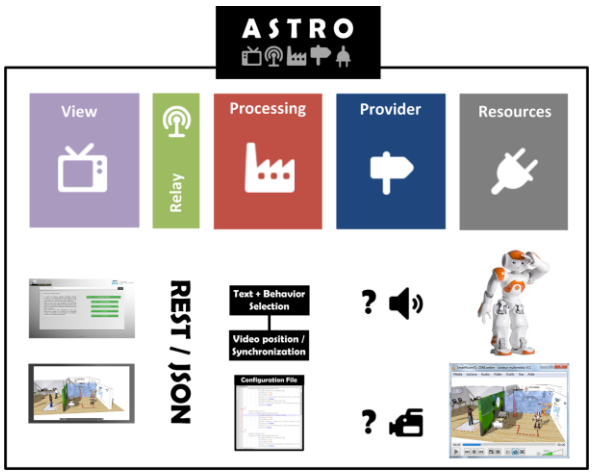

Fig. 2 ASTRO Functional architecture and sample scenario

3. Architecture of Service Toward Robotic Objets

3.1 Overview

As explain in section I, our architecture aims at providing service composition and a framework using robot objects. Moreover, it aims at providing services for robot objects.

We try to cover most of requirements of capabilities management and Service Oriented Middleware as in section II into our different architecture blocs. One of our main goal is to build a modular architecture in order not to only allow a better scaling up but also to be able to embed some blocks directly on smart object (robot, desktop, smart phone). (e.g allowing NAO robot [2] to use ROS ready tools).Giving access to the service and providing to robot the ability to use Web Services is also targeted. Web Compliant communications are available into our architecture.

The ASTRO workflow (figure 2) can be illustrated by the following example. Let assume that we need to use the NAO in order to present a video embedded into our web site. The first step is to create a new Resource, allowing communicating with the NAO. A configuration file will be associated with this resource containing the capabilities of such object (gesture, text to speech, dancing,...).

The second step involves the creation of a Service into the Processing bloc. This service will aim at executing special gesture and text to speech for a given text, and at going to a given video position representing the current text.

Now a Web page should be created allowing user to select a section to be presented. This web page used a javascript library (Web Bloc) that simplifies the

communication with the Relay bloc showing available services through a REST architecture.

When a end-user will select the item on the web page, a request will be sent to the Relay bloc on a specific service. The Relay bloc informs the Processing bloc which transmits the request to the appropriate service. The service calls the Provider bloc to say the selected text, execute a gesture, and freeze the video at a given position.

The Provider bloc researches available resources with such capacities and links service and found resources through a communication BUS (publish/subscriber).

The NAO resource gets the text to speech and the gesture to execute and call the NAO robot to execute the requests.

3.2 Resource Bloc

The Resource bloc aims at delivering an access to capabilities of smart object (figure 3). It is composed of :

- a set of Resources describing the different robot or sensor/actuator drivers and behaviors,
- a Resource Broker module managing the Resources and register them to the Provider Bloc,
- an API allowing to find other resources or services,
- a module in charge of providing communication to the Resource.

User who wants to add a new smart component such as embedded device, sensors/actuators or robot should compose its own Resource and provide the following elements:

- Object Driver, API to the smart object,
- Data Processing, all formatting or processing operations necessary to deliver information or send order,

Resource configuration, configuration file describing the Security information to access to the smart object and to provide authentication information to the Provider bloc, Quality of Service (QoS) informing the Provider bloc of such kind of communication type is needed, capacity description describing all the capacities available with the smart object.

3.3 Provider Bloc

The Provider bloc is in charge of registering Resources and Services using respectively the Resource Broker and the Service Broker (figure 3). By collecting information (Directory module) on resources and services such as capabilities and services usage, the Provider bloc maintains the mapping of identified resources and associated capabilities according to the Ontologies module.

When a service wants to find a capability, the Provider bloc receives the request by the Directory module and gets the result through the Ontologies module (E.G list of resource IDs). The Ontologies module includes the ontologies of capabilities and services. As mentioned in [23], an ontologies allow efficient reasoning on context information and enable service interoperability. Each of these ontologies is plugged to a reasoner in order to provide ontology inference and answer to incoming requests. As soon as elements of response are identified, the Provider bloc establishes communication from request initiator to the selected resources. These communications are managed by the Communication Management module.

When capabilities or services are identified, the Communication Management module connects the service initiator to the targeted resources or services. A Bus Communication is chosen according to the QoS requested by the Service and the Resource. The communication type is publisher subscriber. As explained in [21] and encouraged by in [22] and [1], publisher/subscriber is a very good candidate for resolving scalability issue, reducing redundant messages, administration/establishment cost and decoupling between receiver/provider.

Fig. 4 Astro modular architecture

3.4 Processing Bloc

This bloc is in charge of composing services (custom services) using robot's capacities, sensors/actuator or other services with the help of the Provider bloc (figure 3). The Service module needs to be developed by the service producer. Some configuration information is needed such as security and Qos requested and also the scenario usage

Fig.5. ASTRO Implementation: Web based Scenario of interaction with NAO robot and Video Controller

(as describe into the service ontology III-C). In order to help the developer, a set of API is available. These APIs allow to easily finding capabilities and other services. The Service can deliver information to the Internet through the Relay Communication module that forward information to the Relay Block. The Service broker module manages the Service life-cycle and transmits Service registration to the Provider bloc.

When a service wants to receive information from a capacity or a given service, it calls the provider bloc through the API module, and as soon a communication is established with the associated resources or services, it uses the Astro Communication Module to send order or receive information.

3.5 Relay and Web Bloc

In order to interact with the produced services, the Relay bloc provides a collection of communication to Internet. When a Service is registered, an associated URL is automatically assigned to it in order to use the new Service. Each Service has the capacity to communication through HTTP (REST architecture [23]) for command response communication and through WebSockets for stream communication. The Web bloc, for its part, provides a javascript library that facilitates the usage of the services into HTML web pages.

3.6 Security Concern

In order to guaranty a good behavior of the global system, each Service and Resource is executed into a box. A box manages the life cycle of the component and controls the component activity (Thread Usage, memory consumption) avoiding error and malicious usage. All communication establishments are controlled by the Provider bloc. This bloc will control that services and resources have necessary credential for communicate with other components.

3.7 A modular Architecture

One of the targets of our architecture is to provide a scaling up solution. Communications across several ASTRO architectures are possible. On one hand, the ASTRO architecture could communicate across the Relay Bloc (http request or WebSocket) and uses remote services as other Internet services. On the other hand, the different Provider bloc could be connected each other using specific communication Bus (like JMS [24]). Using the second solution offers to the different architecture the ability to use remote registered capabilities or services. More over, subpart can be embedded directly into smart objects (e.g robot, remote system). If we want to collect information from a Kinect not directly linked to our server but connected to a remote Raspberry Pi, we could install some part of our architecture directly on the remote embedded system shown figure 4.

4. Experimentation

In order to illustrate our architecture, we implement and test the scenario described in I (figure 5). The purpose of the scenario is to provide a web interface to the user that can select topic on a HTML page. Two resources are available, NAO robot with set of capabilities (text to speech, gesture, dancing, face recognition ...) and a video controller that allows displaying information. The video controller holds a single video describing all topics provided into the HTML page. The figure 5 shows the scenario in action. When user select a topic, the ASTRO architecture selects the appropriate service, chooses the targeted resources according to their capacities and then launches the behavior on the NAO robot (Text to speech

and gesture) and on the video controller (pause the video on targeted frame and release video lecture when the NAO finish its speech).

The Web bloc is composed of set of javascript libraries helping to call services (based on jquery). The Relay bloc and services are made above a java Grizzly [25] server (section III) that provides service as a REST architecture. The Services bloc, Provider bloc and Resource bloc are written in Java. For the communication bus, we used the MBassador java Bus (managing message priorities). Currently, we have developed a set of different resources including NAO Robot resource (using NaoQi), ROS resource (currently using RosBridge, a java ROS resource is under development), IP video resource, USB Video resource, Kinect resource (using NITE and Openni) and a general TCP/UDP socket connector. We developed a remote ASTRO agent including Provider bloc and Resource bloc. Current communication are not Bus to Bus (e.g JMS) but done through java Remote Method Invocation communication.

5. Conclusion and Future Work

Robots and smart devices capabilities to web oriented services. Moreover, robots or smart devices can use high level services provided by the Astro architecture.

We evaluate our architecture across the service oriented middleware [4], of robots middleware [5] requirements. Concerning the service oriented Middleware requirements:

- Service composition and standardization: Partially Covered, the service composition is possible by requesting Provider bloc in order to find a service corresponding to the targeted usage. Communications Service to Service are then possible (publisher/subscriber communication). Currently any service standardization is strongly defined.
- Service registry and publishing, Service discovery and integration: Covered, as soon as a service or resource is created, the associated broker registers them to the Provider bloc. Then services and capabilities become available on the ASTRO architecture.
- Heterogeneity: Covered, effort has been made to allow integrating heterogeneous Resources into our architecture. To do so, user needs to develop the object driver. Then the heterogeneity is hidden by the usage of capacities ontology.
- Integration transparency to client applications: Partially covered, using ontology of services and capacities allows to discover Resource or Service without prior knowledge. But the usage of associated resource and service is currently not standard and need a custom implementation.
- Adaptation and autonomicity: Not Covered, Despite that resource of services are found by ontology search, the continuity of service is not supported, the initiator is not dynamically noticed if the service or resource is no more available
- Scalability and efficiency: Partially Covered, our architecture can communicate with other ASTRO architectures and share resources and services, but no benchmark has been currently done on transfer rate and high communication load.
- Reliability and security, QoS requirements: Not currently Covered, the security and QoS of services and resources are specified but no implementation is currently available.

Concerning the robot middleware requirements:

- Simulation: Not Covered, no simulation of workflow is currently available.
- Open Source: Partially Covered, we still work on enhance packaging before sharing our work.
- Behavior coordination: Partially Covered, the behavior coordination of capabilities can be done indirectly by developing a service which coordinates high level actions on robots or smart objects.
- Real Time, Not Covered, the Real Time aspect is not targeted by our architecture.
- Distributed environment, Partially Covered, we provide some tools to communication between different ASTRO architecture through the Relay bloc or the Provider bloc but effort are still needed to provide a complete distributed architecture (Provider bloc Peer to Peer communication).
- Dynamic wiring: Covered, using the ontology of capacities or services, composed service or robot resource can be dynamically wired to suitable service or resource.

The implementation of ASTRO is still in progress. Qos service still need to be implemented. The communication between different ASTRO architecture is currently partially addressed (Remote Method Invocation) and needs to be well defined (services and capabilities across different ASTRO architectures discovery). Moreover, the capacities and service ontologies definition are still in progress and current implementation provides only basic skills (discovery of capabilities and services into predefined lists).

In the current paper we provide an new architecture allowing the composition of service of robot and smart objects. By oriented our work on Web Compliant communication we give the opportunity to open our services to the Web and allow robot to use Web services.

Moreover the user interaction is facilitate by using well known program language (Java, Javascript) and communication protocol(HTTP, WebSockets). The service composition use an abstract robot capacity usage (ontology) that can dynamically ”wired” service to resource (robots, smart devices). Finally, by executing services and resources into boxes and control communication access between service and resource, we provide a first step of robot service securing.

Acknowledgments

The authors would like to thank the contribution of master students, Thomas Bracher and Romain Huet who worked more than 6 months on the implementation of the ASTRO architecture. Authors would like to thank also the Robot Forum incubator of Cpe Lyon to have integrated this project into its smart room.

References

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in ICRA Workshop on Open Source Software, 2009.

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in ICRA Workshop on Open Source Software, 2009.

[2] Aldebaran robotics. [Online]. Available: http://www.aldebaranrobotics.com

[3] M. Waibel, M. Beetz, R. D’Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. G´alvez-L´opez, K. H¨aussermann, J. Montiel, A. Perzylo, B. Schießle, O. Zweigle, and R. van de Molengraft, “Roboearth - aworld wide web for robots,” Robotics & Automation Magazine, vol. 18,no. 2, pp. 69–82, 2011.

[4] J. Al-Jaroodi and N. Mohamed, “Service-oriented middleware: Asurvey.” J. Network and Computer Applications, vol. 35, no. 1, pp.211–220, 2012.

[5] A. Y. Elkady and T. Sobh, “Robotics middleware: A comprehensive literaturesurvey and attribute-based bibliography,” Journal of Robotics, vol. 2012, 1 2012.

[6] D. Bonino and F. Corno, “Domains: Domain-based modeling forambient intelligence.” Pervasive and Mobile Computing, vol. 8, no. 4,pp. 614–628, 2012.

[7] N. Ibrahim and F. L. Moul, “A survey on service compositionmiddleware in pervasive environments,” CoRR, vol. abs/0909.2183, 2009.

[8] Webots, “http://www.cyberbotics.com,” commercial Mobile RobotSimulation Software. [Online]. Available: http://www.cyberbotics.com

[9] O. Michel, “Webots: Professional mobile robot simulation,” Journal of Advanced Robotics Systems, vol. 1, no. 1, pp. 39–42, 2004.

[10] C. Schlegel and R. Worz, “The software framework smartsoft forimplementing sensorimotor systems,” in Intelligent Robots and Systems,1999. IROS ’99. Proceedings. 1999 IEEE/RSJ International Conference on, vol. 3, 1999, pp. 1610 –1616 vol.3.

[11] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, “Rtmiddleware:distributed component middleware for rt (robot technology).”in IROS. IEEE, 2005, pp. 3933–3938.

[12] S. Magnenat, V. Longchamp, and F. Mondada, “Aseba, an event-basedmiddleware for distributed robot control,” in Workshops and TutorialsCD IEEE-RSJ 2007 International Conference on Intelligent Robotsand Systems. IEEE Press, 2007.

[13] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, “Miro -middleware for mobile robot applications,” Robotics and Automation,IEEE Transactions on, vol. 18, no. 4, pp. 493–497, Dec. 2002.

[14] H. Bruyninckx, P. Soetens, and B. Koninckx, “The real-time motioncontrol core of the Orocos project,” in IEEE International Conferenceon Robotics and Automation, 2003, pp. 2766–2771.

[15] G. B. T. S. H. N. T. K. Noriaki Ando, Shinji Kurihara, “Softwaredeployment infrastructure for component based rt-systems,” Journalof Robotics and Mechatronics, vol. 23, pp. 350–359, 2011.

[16] “Robotic interaction service (rois),” Object Management Group(OMG), 2013. [Online]. Available: http://www.omg.org/spec/RoIS/1.0/

[17] C. Crick, G. Jay, S. Osentoski, and O. C. Jenkins, “Ros and rosbridge:Roboticists out of the loop,” in Proceedings of the Seventh AnnualACM/IEEE International Conference on Human-Robot Interaction,ser. HRI ’12. New York, NY, USA: ACM, 2012, pp. 493–494

[18] M. Bichler and K.-J. Lin, “Service-oriented computing,” Computer,vol. 39, no. 3, pp. 99–101, 2006.

[19] P. Jain, D. Dahiya, and W. Solan, “An architecture of a multi agententerprise knowledge management system based on service orientedarchitecture,” IJCSI International Journal of Computer Science Issues,vol. 9, pp. 395–404, 2012.

[20] J. Anke and al., “A service-oriented middleware for integration andmanagement of heterogeneous smart items environments,” in 4thMiNEMA Workshop, 2006, pp. 7–11.

[21] C. Groba, I. Braun, T. Springer, and M. Wollschlaeger, “A serviceoriented approach for increasing flexibility in manufacturing.” Proceedingsof 7th IEEE International Workshop on Factory CommunicationSystems, COMMUNICATION in AUTOMATION (WFCS 2008),Dresden, Germany, 2008.

[22] D. Bonino and F. Corno, “Dogont - ontology modeling for intelligentdomotic environments,” in Proceedings of the 7th International Conferenceon The Semantic Web, ser. ISWC ’08. Berlin, Heidelberg:Springer-Verlag, 2008, pp. 790–803.

[23] H. Guermah, T. Fissaa, H. Hafiddi, M. Nassar, and A. Kriouile, “Anontology oriented architecture for context aware services adaptation,”IJCSI International Journal of Computer Science Issues, vol. 11, pp.24–33, 2014.

[24] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, “Themany faces of publish/subscribe,” ACM Comput. Surv., vol. 35, no. 2,pp. 114–131, Jun. 2003.

[25] P. R. Pietzuch and J. Bacon, “Hermes: A distributed event-basedmiddleware architecture,” in Proceedings of the 22Nd

InternationalConference on Distributed Computing Systems, ser. ICDCSW '02.Washington, DC, USA: IEEE Computer Society, 2002, pp. 611–618.
[26] R. T. Fielding, "Rest: Architectural styles and the design of networkbasedsoftware architectures," Doctoral dissertation, University ofCalifornia, Irvine, 2000.
[27] R. Monson-Haefel and D. Chappell, Java Message Service. Sebastopol,CA, USA: O'Reilly & Associates, Inc., 2000.
[28] Project grizzly. [Online] Available: https://grizzly.java.net/overview.html

Jacques Saraydaryan holds a Master's Degree in Telecoms and Networks from National Institute of Applied Sciences (INSA), Lyon –France in 2005, and a Ph.D. in computer sciences from INSA, Lyon France in 2009. He was a Research Engineer at the Exaprotect Company, France during seven years. His research focus began on IS Security especially on Anomaly intrusion detection system (published in international conferences such as Secrureware'08, SEC 2008). Currently, Jacques is a member of the engineers School of CPE Lyon as associate professor. His experience into behavior modeling helps him to integrate the robotic research team of the school.

Fabrice Jumel is associate professor in computer science and robotic of service at the CPE Lyon Engineering School. He received its Ph.D. degree from "Mines de Nancy" Engineering School and the INRIA (TRIO team) in 2003. Its dissertation was about "quality of service of control systems". His researches focus on biomedical systems, domotic systems and robotics. He is a the head of the robot forum initiative of CPE Lyon in which they build, improve and work on a smart experimentation area. This area includes a lot of robotic solutions, Human Machine Interfaces, cameras, depth cameras, light, heat controllers and biomedical devices.

Adrien Guenard holds a Master's Degree in control desgin and robotics from ENSEM engineering school. Adrien was a Research Engineer at INRIA, working on models and simulation of wireless sensor networks and mobile platforms during 2 years. After that, he worked at the Loria laboratory as a multirotor UAV expert. He joined the robotics research team of CPE school in 2012.