Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 94 (2016) 121 – 128

Procedia Computer Science

The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016)

An agent middleware for supporting ecosystems of heterogeneous web services

Pablo A., Pico-Valencia[a], Juan A. Holgado-Terriza[b]*

[a]*Pontificia Univ. Católica Ecuador (Esmeraldas), Esmeraldas 080150, Ecuador*
[b]*Universidad de Granada, Granada 18071, Spain*

Abstract

The integration of a Multi-agent technology with a service oriented architecture provides a convenient way to build smarter applications that satisfy the demand of the current ubiquitous web systems. This paper provides a software tool to develop ubiquitous applications adopting the philosophy of agents and services as data sources. ADELE (Agent Dynamic EvoLutionary at runtime) is a middleware that allows developers to create reactive agents with the capability to evolve through the injection of external behaviors at runtime. An ADELE external behavior is a software component that allows agents to accomplish their goals. To facilitate the programming of these behaviors, agents can obtain the information consuming local and public web services (WSs) previously published on different services ecosystems. This paper shows how a Multi-agent System can consume heterogeneous WSs to satisfy the agent goals using a normative model. We have created three add-ons compatible with SOAP, RESTful, and DOHA (Dynamic Open Home-Automation) WS model. The integration of these add-ons within the ADELE tool is helpful to facilitate the invocation of heterogeneous WSs with a high abstraction level. In addition, we describe as an example, an Internet of Thing (IoT) scenario where the approach presented in this paper is very helpful. Finally, we also evaluate an ADELE application for giving home comfort employing heterogeneous WSs.

Keywords: Multi-agent; ADELE; web service; SOAP; RESTFul; DOHA; service ecosystem; dynamic client;

* Corresponding author. Tel.: +34-958-240-570; fax: +34-958-243-179.
E-mail address: jholgado@ugr.es

doi:10.1016/j.procs.2016.08.020

1. Introduction

New trends emerging around the Internet technologies are changing the way about how web and mobile applications can offer new-value services to users in order to satisfy their needs. The use of semantic models to achieve interoperability[1], the inclusion of agents to add intelligence and mobility[2] or the application of Cloud Computing to share resources and trying everything as a service[3], are some trends in the development of current web and ubiquitous applications. However, one of the most promising technologies in these fields is undoubtedly related to Web Services (WSs), since they are able to create interoperable communications between machines over the Internet[4] managing also the requests of billions of web and devices that are interconnected at any time everywhere. This is the origin of the current Internet of Service (IoS).

The IoS describes an infrastructure through a logical collection of WSs which uses the Internet as a medium to offer, change, adapt, sell and operate WSs for any consumer using one or more WS ecosystems[5]. Considering that a WS ecosystem can flood Internet with many atomic and composite services, it is required to apply a discovery stage for finding services and an execution stage for describing how the realization of a service is carried out[5]. In this line, several research works have proposed the fusion between Service Oriented Architecture (SOA) and Multi-agent Systems (MASs)[6,7,8]. However, we have considered to use the approach defined by Paz[9] that exposes agent behaviors as WS (SOAP services) without using a discovery process; but, in our approach, this idea is extended in order to support the integration of heterogeneous WSs in agent behaviors. In fact, we can access to SOAP (Simple Object Access Protocol)[10], RESTful (Representational State Transfer)[11], and DOHA (Dynamic Open Home Automation)[12] WSs to compose basic and complex functionalities from these ones, and indirectly helps agents to meet their goals.

Though WSs have important implicit features for the development of scalable distributed systems such as the autonomy, the simplicity, and the interoperability, they do not have the ability to act intelligently. However, the fact that atomic and composite WSs can be encapsulated within the behavior of agents at compile time or at runtime, we can achieve a higher level of interoperability at MASs level. This is the main reason why we propose a middleware able to build agents that satisfying their goals by using data obtained from heterogeneous WSs. We have also considered the use of several WS technologies, such as SOAP, RESTful and DOHA. SOAP and RESTful were selected because they are currently the most widespread WS technologies for the development of this type of software components[13]. In addition, we have also included a specific WS technology, DOHA[14] based on DPWS¹⁵¹⁵ (Device Profile Web Service), because in ubiquitous scenarios, it may require lightweight WSs which act on scenarios with limited resources[15].

The future of the Internet and even the future of the web are mainly focused on building smarter applications with the capacity to satisfy the requirements of the future ubiquitous web[16]. In applications of this nature the mobility and adaptation are two elementary aspects[17] that they are still under investigation. Respect to the adaptation we have used the ADELE (Agent Dynamic EvoLutionary at runtimE) middleware[18] which offers the adjustment of agents by means of the injection of external behaviors at runtime for giving to developers the capacity to build these external behaviors by using Dynamic Clients (DC) or add-ons that allow invoking directly heterogeneous WSs that belong to distinct service ecosystems. Thereby, ADELE becomes a useful software tool for creating agent platforms that can operate intelligently on ubiquitous scenarios thanks to its mechanisms of evolutions at runtime and the capability to work with WSs independently of their location and technology.

The content of this paper is structured in five sections. In Section 2 we discuss some of the related works. Section 3 introduces the main concepts associated with the basic unit of the IoS. Section 4 presents the architectural model of ADELE to support agent actions as a composition of heterogeneous WSs. In addition, a description of the add-ons SOAPDC, RESTDC and DOHADC are described. In Section 5, we discuss the obtained results of the evaluation of an ADELE application for giving comfort to a smart home that includes agent goals which are accomplished using the composition of heterogeneous WSs, and we compared with other similar application that solves the same agent goals, but by using each service model individually. Finally, conclusions are commented.

2. Related works

Some years ago the approaches related to services and agents have been worked as independent self-contained technologies without any interaction. However, in the last decade, the integration of both approaches has widely been accepted by their complementary features for building distributed, open and flexible applications, which have led to smarter applications that satisfy the demand of the current ubiquitous web. As a result of the integration of the MAS and SOA into a MAS-SOA approach, the term "smart service" was defined to give an answer to the needs of current applications in relation to the autonomy, flexibility, adaptability and interoperability[19].

MAS-SOA integration has been conceived in different ways. In [6] Tapia proposed an architecture to facilitate the integration of distributed services into a MAS without including the functionality inside the agent structure. To achieve the MAS-SOA integration it uses a service directory to publish the services that an agent offers to others. In [7], Ry and Radziszewska also presented a hybrid architecture for the MAS-SOA integration, but in this case the mechanism is focused into the communication model that allows interaction with external applications using add-ons such as WSDC (Web Service Dynamic Client). But it is only limited to consume SOAP services. In the same way Herrera presented in [8] an architecture named Service Oriented Cross layer infRAestructure for Distributed smart Embedded devices (SOCRADES) to integrate MAS and SOA in industrial automation. It uses a Decision Support System (DSS) process to select the adequate services before doing the invocation. But also it is limited for invoking DWPS services. In [20], Fernandez-Villamor presented an architecture for the discovery of RESTful services and content on the web using Belief-Desire-Intention (BDI) agents that perform plans based on the induction of rules for discovery in a REST architecture style. However, other mechanisms for merging MAS and SOA that do not make use of a service discovery process is presented by Paz, in [9]. He considered that actions of agents can be exposed as functional or process WSs for performing complex tasks by means of additional components such as WSIG (Web Services Integration Gateway) or WS2JADE (Web services in Java Agent Development Framework).

As we can note, the architectures proposed for the MAS-SOA integration at present, are focused to invoke only an individual WSs category between the following ones: SOAP, RESTful, DPWS or others. The selection of one of these categories is not a simple task and mainly depends on the nature of the application to develop. Hence, instead of choosing a specific category of WSs to develop ubiquitous smart applications, the software analyst ought to consider the evaluation of performance between WSs based on SOAP and RESTful. Regarding this issue many works[13,21,22] had evaluated the performance between these two categories of WSs. In general the results show that RESTful WSs have better response time compared to SOAP services[21,23], because they consume less amount of memory in contrast with SOAP WSs[13], and the size of the messages are relatively smaller than in SOAP WSs[22].

For getting the interoperability in a system, it requires the inclusion of mechanisms that allow the communication with external systems. To achieve it, at SOA level, it is possible to create compositions of WSs starting from atomic and composite services by adopting a specific service composition model. A review of the main mechanisms is presented in [24,25] to compose SOAP services that include composition models based on orchestration, choreography, workflow or planning; methods to deploy service composition in a manual, semi-automatic or automatic way; or different strategy to make up the composition based on a static or dynamic selection of WSs.

These composition models only can be carried out by using mainly the Business Process Execution Language (WS-BPEL) standards, although it is also possible to achieve it by using other languages such as Web Services Choreograph Description Language (WS-CDL), Business Process Modelling Language (BPML), Semantic Markup for Web Services OWL-S or Web Service Modelling Framework (WSMF). However, many of these standards are only compatible with SOAP services and not to support heterogeneous WSs. That is the reason because the WSDL 2.0 language can already describe RESTful services and therefore also could be composed[26] by means of BPEL or using RESTML as proposes Vieira in[27]. Other approaches as Rodriguez[12] presented a high level service composition model based on DOHA services to improve the data handling managed by devices in IoT scenarios by using device services that encapsulate the functionality and restrictions of physical devices. In contrast, Garriga[26] proposed a similar mechanism for composing RESTful instead of SOAP WSs. However, few works explore directly the composition of heterogeneous WSs. Thus, Lee[28] proposed the composition of heterogeneous WSs combining SOAP, non-SOAP and non-web services in order to define complex tasks.

3. ADELE and MAS-SOA approach for IoT

3.1. Internet of Things and Internet of Services

An Internet of Things (IoT) scenario is a modern wireless environment where there is a pervasive presence of electronic things or smart objects which are able to interact with others, sharing information, cooperating with nearest devices, and coordinating decisions[29]. To achieve the basic functionality required by this kind of scenarios, Al-Fuqaha specifies that the delivery of IoT applications needs to define a set of six main elements such as identification, sensing, communication, computation, services, and semantic[30]. Therefore, IoS plays an important role in IoT applications because it provides mechanisms that allow people to use devices and smart objects which perform actions by means of WSs belonging to one or more service ecosystems[31].

Since services are one of the main six elements considered by an IoT architecture, being the WSs the most natural way to implement them, they turn into a very important component in IoT scenarios. Currently there are several models such as SOAP, RESTful, DPWS, DOHA, among others, which offers different features.

3.2. MAS-SOA approach and heterogeneous WSs

IoT scenarios require normally certain level of intelligence that optimize the acquisition and management of data resources and devices connected to the network any time at everywhere. This is the main reason why the integration of services and agents could be an important challenge.

An architecture that fuses MAS and SOA requires at first instance components that allow consuming WSs already implemented. In this regard, we have implemented three add-ons compatible with SOAP, RESTful and DOHA as is shown in Fig. 1. These three service models were selected because they covers the WS technology spectrum with important features that are useful for IoT scenarios.

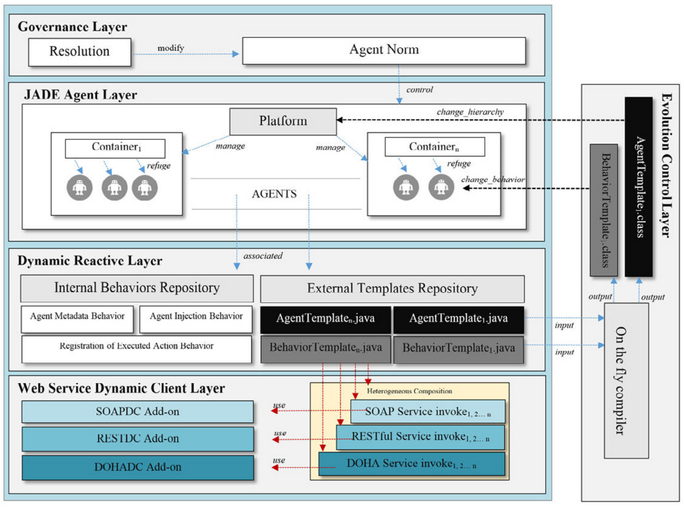

Fig. 1. ADELE architecture and SOAP, REST, and DOHA Dynamic Clients.

Concerning to the consumption service, the main process performed by each DC is based on request-response communication process. In the case of SOAP WSs, the client consumes the operations included in the service contract implemented through WSDL. On the other hand, in the RESTful model the request is made using a Uniform Resource Identifier (URI), an HTTP method (GET, PUT, POST, DELETE) and a type of message representation (XML, JSON, among others) to access to the requested resource deployed by the REST server[4], Finally, the DOHA service model explores devices internally for finding services[12], although the access to other services is carried out by means of the URL of their service contracts, also described in WSDL, but is standard for DPWS services. Next, we describe each add-on and the stages for use each one of them.

- SOAP Dynamic Client (SOAPDC): For invoking SOAP WSs, we have developed a DC following the basis of WSDC[32]. The first required step is to create a new instance of *dynamicclients.SoapDC*. The instance is initialized by using a method named *initialize()* employing the information recovered from a valid WSDL service contract accessible through an URL of the specific SOAP WS. Then, the service can be called by using the method *invokeServiceSTRING()* or *invokeServiceJSON*() including the specific operation and the list of parameters and their values. This DC accomplishes the request-response by the exchanging of SOAP messages between DC and the WS provider.
- REST Dynamic Client (RESTDC): To consume RESTful WSs, we have developed a new DC similar to SOAPDC. The instantiation is done from *dynamicclients.RestDC* similarly to above. It does not require an initialization stage, but it invokes directly the service sending a valid URI, the resource requested and the value

or a set of values using a list as parameters. The response is delivered as a String object or a JsonObject.

- DOHA Dynamic Client (DOHADC): The execution of the DOHA WSs is carried out by the instantiation of *dynamicclients.DohaDC*. Then, the execution of the method *initialize()* gives access to devices and their running services on the subnet through passing the URL of the WSDL service contract as parameter. Then, the invocation can be executed through the method *invokeServiceSTRING()* or *invokeServiceJSON()* passing the specific operation, a list of parameters, and their values. The results of requests are delivered as a String or a JsonObject.

Regarding to the MAS side, we have applied the ADELE middleware basis[18]. Fig. 1 also shows the four primitive layers (governance, dynamic reactive, evolution control and JADE agents) that compose the architecture of this tool. However, we have added a new layer named *Add-ons Dynamic Client Layer* that allows the accomplishment of the agent goals by accessing to heterogeneous WSs. Therefore, our approach is able to evolve based on a normative model (*Governance Layer*) formed by norms that affect one or more active agent instances (*JADE Agent Layer*). Each norm is linked to one external behavior which is introduced within the MAS at runtime and injected to active agents affected by the norm associated with the behavior (*Evolution Control Layer)*. Thus, when an adaptation is required, it is possible to add new norms and the corresponding behavior without stopping the system. The *Add-ons Dynamic Client Layer*, integrates three disengaged add-ons that include methods for invoking SOAP, RESTful and DOHA WSs from an external behavior template. MAS-SOA integration is carried out here, but adopting the criteria proposed by Lee[28] where the composition of complex actions is performed by means of heterogeneous services and not limited only to one service category as is presented in [6,7].

4. Evaluation

4.1. Definition of a scenario

We have considered a scenario where a smart home keeps maintaining the level comfort to the inhabitants. The level of comfort can be achieved by regulating some specific parameters (e.g., light level, temperature or humidity) individually or collectively. The smart home system has to look out that the comfort parameters are accomplished and should activate automatically the corresponding actuating devices (e.g., air conditioning system (HVAC), heating) when the systems is outside of the user consigns.

For accomplishing the above requirements in ADELE, we have selected some specific parameters such as the temperature or the humidity. However the system can be scaled adding new parameters or by fusing the information between some measured parameters. Fig. 2 illustrates the components of the system.

The development of the smart home system has followed the next stages: (i) installation and deployment of sensors and devices (temperature and humidity sensors, HVAC system and actuator controller), (ii) creation of all the WSs required to get access to temperature, humidity and HVAC regulator, and a WS named Historize for storing data of any measurement with a timestamp, (iii) searching of public WSs that provide meteorological data (e.g., *YahooWeatherService* based on RESTful model), (iv) creation of the agents such as temperature, humidity, HVAC controller and comfort agents, (v) creation of the norms, i.e. from N_1 to N_4, and finally, (vi) the creation of the external behaviors linked to the created norms that invoke the WSs at service layer.

In addition, Fig. 2 also shows an illustration of the basic pieces of ADELE middleware once the physical devices have been deployed. According to (ii) we developed the *TemperatureService* and the *HVACControllerService* based on the DOHA model, the *HumidityService* employed a SOAP model, and the *HistorizeService* with a RESTful model. In this way we can contrast the achieved results by using heterogeneous WSs. All these WSs were developed in Java using DOHA middleware[14] in the case of DOHA WSs, JAX-WS (Java API for XML WSs) for SOAP WSs, and JAX-RS (Java API for RESTful WSs) for RESTful WSs. The last two WSs were deployed on a local Glassfish sever to avoid a possible dependency with the Internet connectivity.

Once the WSs were deployed, we can adjust the execution of ADELE in base of the defined requirements in the normative. Therefore, an ADELE generic agent was created using an agent template with the same basic behavior for each required agent type (*TemperatureAgent*, *HumidityAgent, HVACControllerAgent,* and *ComfortAgent*). As a consequence of this, ADELE could compile automatically these templates and could generate the corresponding executable in order to create the agent instances. The adoption of a generic behavior emplaces the assumption of a specific base behavior such as the support of the injection of external behaviors.

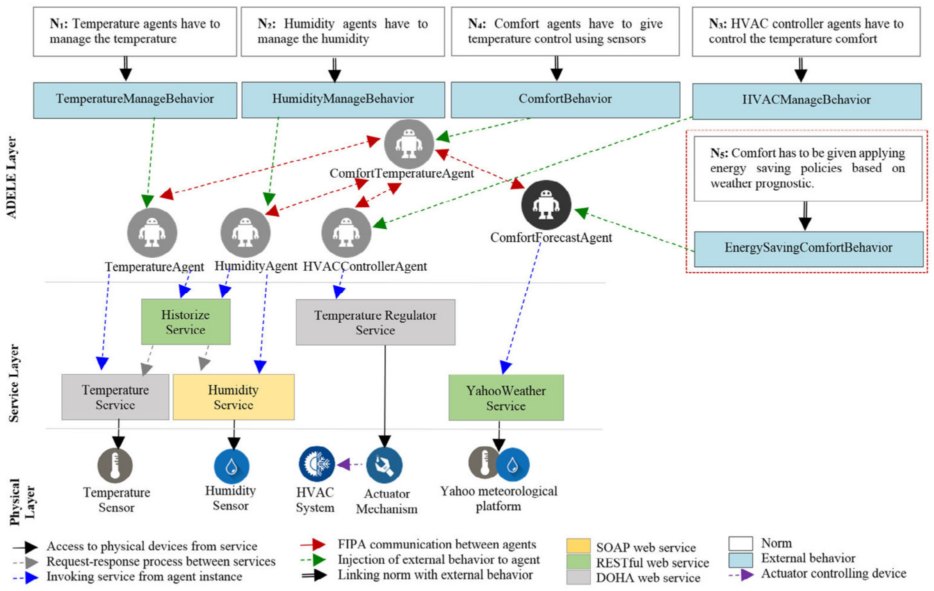
N1: Temperature agents have to manage the temperature
N2: Humidity agents have to manage the humidity
N4: Comfort agents have to give temperature control using sensors
N3: HVAC controller agents have to control the temperature comfort
TemperatureManageBehavior
HumidityManageBehavior
ComfortBehavior
HVACManageBehavior
N5: Comfort has to be given applying energy saving policies based on weather prognostic.
EnergySavingComfortBehavior
ADELE Layer
ComfortTemperatureAgent
ComfortForecastAgent
TemperatureAgent
HumidityAgent
HVACControllerAgent
Service Layer
Historize Service
Temperature Regulator Service
Temperature Service
Humidity Service
YahooWeather Service
Physical Layer
Temperature Sensor
Humidity Sensor
HVAC System
Actuator Mechanism
Yahoo meteorological platform
Access to physical devices from service
Request-response process between services
Invoking service from agent instance
FIPA communication between agents
Injection of external behavior to agent
Linking norm with external behavior
SOAP web service
RESTful web service
DOHA web service
Norm
External behavior
Actuator controlling device

Fig. 2. Smart home scenario for adding service of comfort of temperature using ADELE middleware and heterogeneous WSs.

Each agent instance should have a behavior that is not yet defined. However, next we will assign their expected behavior as is shown in Fig. 2. In the case of the *TemperatureAgent* it should achieve temperature values searching a specific WS from the yellow pages, while the *HumidityAgent* looks for a specific WS to get humidity values. Similarly, the *HVACControllerAgent* could find regulation strategies on yellow pages to apply to an actuator system in order to keep a temperature consign, while the *ComfortAgent* could give comfort from values given by both *TemperatureAgent* and *HumidityAgen*t. It is possible to have a cooperative behavior; e.g., the *ComfortAgent* could profit the values of temperature, humidity, or another parameter that other agents may share with it.

The agents can select at any moment the required WSs from a list of available WSs from a catalog of yellow pages when the agent need the resources or functionality encapsulated in a WS o several WSs to accomplish their goals. In the case of *TemperatureAgent* we registered the *TemperatureService*, for *HumidityAgent* the *HumidityService,* and finally, for *HVACControllerAgent* the *HVACControllerService*. We also registered the *HistorizeService* for both *TemperatureAgent* and *HumidityAgent*.

Later, we needed to create the ADELE norms and their corresponding external behaviors. The norms (N_1-N_4) are created from user requirements. The norm N_1 defines that the agent instance created from the *TemperatureAgentTemplate* has to get the current value of temperature, while N_2 specifies that the agent instance built from *HumidityAgentTemplate* has to obtain the current value of humidity. On the other hand, the norm N_3 establishes the obligation for the agent instance created from the *HVACControllerAgentTemplat*e to control the actuator mechanism for handling the HVAC system according to the *ComfortAgent* requests. Finally N_4, defines the agent instance created from *ComfortAgentTemplate* that has to the allow communication process with the three agents already detailed or whichever one that can cooperate with data for carrying out the giving comfort process.

After the creation of the norms, we linked each one to an external behavior. In the case of the *TemperatureManageBehavior* and *HumidityManageBehavior* we developed a mechanism to search on yellow page the WSs that give the temperature and humidity parameters, respectively. The *HVACManageBehavior* encapsulates the required functionality for finding WSs that provides resources or devices which can modify the room temperature. Because we previously registered the *TemperatureService, HumidityService, HVACControllerService,* and *HistorizeService* on the yellow pages, the *TemperatureAgent* is capable to find the *GetTemperature()* and *Historize()* operations, the *HumidityAgent* can locate the *GetHumidity()* and *Historize()* operations, and *HVACControllerAgent* also can discover the *SetHVACTemperature()* operation. On the other hand, *ComfortBehavior* enclosed a different behavior which involves *TemperatureAgent* and *HumidityAgent* for obtaining value parameters, and *HVACControllerAgent* to regulate the room temperature. In this case *ComfortBehavior* set FIPA communications with the corresponding agents.

With the accomplishment of these updates we solved the whole user requirements, adapting the smart home system at runtime applying the MAS-SOA approach and consuming WSs of different services ecosystems.

In order to test the ADELE benefits we changed the active norms of the smart home adding a new norm. This allows to verify if ADELE can adapt automatically to the new imposed requirements from the active norms. For example, we have considered an evolving process that change the operating mode of the system already built. In this case, the new user requirement consists in a system adaptation to adopt an energy saving policy defined in the norm N_5 whose functionality is encapsulated on the *EnergySavingComfortBehavior*. This policy involves the regulation of the HVAC system according to the weather forecast and not only using the data measured by sensors. To accomplish this new requirement, we added a new agent named *ForecastAgent* that obtains weather data such as temperature, humidity, rains or snowfall, from *YahooWeatherService*. Then, this agent can inform in advance to the *ComfortAgent* that controls the HVAC system in order to avoid turning on the system on peak hour or just when it is very cold or very hot. On conclusion the system can change the overall behavior influenced by the execution of the active agents in order to satisfy the active norms according to the current user requirements. The automatic adaptations of the system can also occur when the agent cannot access to specific WS or the agent can search a new WS on the catalog.

4.2. Analysis of results

The results shown in Table 1 were obtained after five executions of each WS composition type in the same scenario. We employed a Toshiba computer with an i5 2.3GHz processor, 4GB of RAM, 64 bits Windows 8.1 Operating System, Java SE 1.8.0, JADE v4.3 and Glassfish 4.1 server to deploy WS. The RESTful composition needed an average execution time of 33.77 ms, DOHA composition required 55.53 ms and, finally, SOAP composition needed 76.50 ms. These results confirm the judgment about the better performance of RESTful WSs compared with SOAP WSs according to [13,21,22,23,33,34]. However, DOHA provides results of execution time between RESTful and SOAP, because the DOHA exploits the light WSs used by DPWS WSs that are executing under DOHA. Moreover, after using the heterogeneous compositions using SOAP and DOHA for accessing of humidity and temperature, respectively, and RESTful for historicizing, we required 40.16 ms. This is, 48% more efficient than SOAP behavior, 27% better than DOHA behavior, but 19% worse than RESTful behavior. However, not all providers offer only RESTful WSs and therefore, our proposal can be a good approach when the functionality or resource is already implemented by other service model based on SOAP or DOHA.

Table 1. Average execution time obtained of the evaluation of SOAP, RESTful, DOHA and heterogeneous behaviors (HB).

Times Required (TR)	SOAP behavior (*ms*)	RESTful behavior (*ms*)	DOHA behavior (*ms*)	Heterogeneous b. (*ms*)
TR to Get Temperature	9,83	4,32	5,00	5,00
TR to Get Humidity	10,83	5,11	5,03	10,83
TR to Historize Data	55,84	24,33	45,50	24,33
Total TR	76,50	33,77	55,53	40,16

5. Conclusions

The current technological advances require interoperable applications that take in account metrics as less time for execution, less network traffic, light technologies, and whatever that cooperates with their best performance. Our approach facilitates the creation of interoperable applications capable to support heterogeneous WSs (SOAP, RESTful, and DOHA) belonging to different services ecosystems for the accomplishment of user requirements. In addition, this approach allows the agent and system adaptation at runtime based on a normative model by using metaprogramming techniques that helps to the inclusion of new components (external behaviors and agent classes) by means of the compilation "on the fly" available in Java language.

ADELE is a helpful tool that allows to build adaptable ubiquitous applications based on the MAS-SOA and the normative approach. The inclusion of new user requirements implies to change the active norms in order to perform an external behavior injection o an injection of a new agent class, without stopping the entire system. Nonetheless, we need to do some efforts related with the behavior programming as well as the behavior testing individually or globally, analyzing the impact that a new norm could originate to agent and MAS level.

Acknowledgements

This work was funded by the Ecuadorian Ministry of Higher Education, Science, Technology and Innovation (SENESCYT) through the Program of PHD for university professors.

References

1. Krishnaswamy S, Way F. The Ubiquitous Semantic Web: Promises, Progress and Challenges. 2014;10(December):1-16.
2. Chang YS, Fan CT, Juang TY. Supporting software intelligence in ubiquitous environment exploits mobile agent. *J Ambient Intell Humaniz Comput*. 2012;3(2):141-151.
3. Duan Y, Cao Y, Sun X. Various " aaS " of Everything as a Service. 2015.
4. Zack Shelby. Embedded web services. 2010;(December):52-57.
5. Cardoso J, Voigt K, Winkler M. Service engineering for the Internet of services. *Enterp Inf Syst 10th Int Conf ICEIS 2008*. 2009:15-27.
6. Tapia DI, Rodríguez S, Bajo J, Corchado JM. FUSION@, A SOA-Based Multi-agent Architecture. 2009:99-107.
7. Ry D, Radziszewska W. Integration between Web Services and Multi-Agent Systems with Applications for Multi-commodity Markets. 2012:65-77.
8. Herrera V, Bepperling A, Lobov A, Smit H, Colombo W, Lastra JLM. Integration of Multi-Agent Systems and Service-Oriented Architecture for industrial automation. *IEEE Int Conf Ind Informatics*. 2008:768-773.
9. Pablo J, Grau P, Sanz AC. An Evaluation of Integration Technologies to Expose Agent Actions as Web Services. 2014;279:259-270.
10. Gudgin M, Hadley M, Mendelsohn N, et al. SOAP Version 1.2 Part 1: Messaging Framework. 2007. https://www.w3.org/TR/soap12-part1/. Accessed January 30, 2016.
11. Roy Thomas Fielding. Architectural Styles and the Design of Network-based Software Architectures. *PhD Thesis*. 2000.
12. Rodríguez-Valenzuela S, Holgado-Terriza JA, Gutiérrez-Guerrero JM, Muros-Cobos JL. Distributed service-based approach for sensor data fusion in iot environments. *Sensors (Switzerland)*. 2014;14(10):19200-19228.
13. AlShahwan F, Moessner K. Providing SOAP web services and RESTful web services from mobile hosts. *5th Int Conf Internet Web Appl Serv ICIW 2010*. 2010:174-179.
14. Rodríguez-Valenzuela S, Holgado-Terriza JA, Petkov P, Helfert M. Modeling Context-Awareness in a Pervasive Computing Middleware Using Ontologies and Data Quality Profiles. *Commun Comput Inf Sci*. 2013;413 CCIS:271-282.
15. Hilbrich R. An Evaluation of the Performance of DPWS on Embedded Devices in a Body Area Network. *2010 IEEE 24th Int Conf Adv Inf Netw Appl Work*. 2010:520-525.
16. Sheng QZ, Shakshuki EM, Yu J. Ambient and context-aware services for the future web. *Comput J*. 2015;58(8):1687-1688.
17. Nixon L, Hench G, Lambert D, Filipowska A, Simperl E. The Future of the Internet of Services for Industry: the ServiceWeb 3.0 Roadmap.
18. Pico-Valencia P, Holgado-Terriza JA. ADELE: A middleware for supporting the evolution of multi-agents systems based on a metaprogramming approach. *14th Int Conf Pract Appl Agents Multi-Agent Syst*. 2016;in press.
19. Zhou HJ, Cao JZ, Guo CX, Qin J. The architecture of intelligent distribution network based on MAS-SOA. *Power Energy Soc Gen Meet 2010 IEEE*. 2010:1-6.
20. Fernández-Villamor JI, Iglesias CA, Garijo M. A framework for goal-oriented discovery of resources in the RESTful architecture. *IEEE Trans Syst Man, Cybern Syst*. 2014;44(6):796-803.
21. Kumari S, Rath SK. Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration. *2015 Int Conf Adv Comput Commun Informatics*. 2015:1656-1660.
22. Hamad H, Saad M, Abed R. Performance evaluation of restful web services for mobile devices. *Int Arab J e-Technology*. 2010;1(3):72-78.
23. Mohamed K, Wijesekera D. Performance analysis of web services on mobile devices. *Procedia Comput Sci*. 2012;10:744-751.
24. Taylor P, Garriga M, Flores A, et al. Web Services Composition Mechanisms: A Review. 2016;4602(March 2015):37-41.
25. Sheng QZ, Qiao X, Vasilakos A V., Szabo C, Bourne S, Xu X. Web services composition: A decade's overview. *Inf Sci (Ny)*. 2014;280:218-238.
26. Garriga M, Mateos C, Flores A, Cechich A, Zunino A. RESTful Service Composition at a Glance: a Survey. *J Netw Comput Appl*. 2015;60:32-53.
27. Vierira R, Ramos R, Pontin R. REST: Advanced Research Topics and Practical Applications. In: *Springer*. Vol ; 2014:69-89.
28. Lee J, Lee S-J, Wang P-F. A Framework for Composing SOAP, Non-SOAP and Non-Web Services. *IEEE Trans Serv Comput*. 2014;PP(99):240-250.
29. Shang X, Zhang R, Zhu X, Zhou Q. Design theory, modelling and the application for the Internet of Things service. *Enterp Inf Syst*. 2015;7575(August 2015):1-19.
30. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Commun Surv Tutorials*. 2015;PP(99):1-1.
31. Eloff JHP, Eloff MM, Dlamini MT, Zielinski MP. Internet of People, Things and Services - The Convergence of Security, Trust and Privacy. In: *Third International Workshop IoPTS*. Vol ; 2009.
32. Caire G. *Web Services Dynamic Client Guide*. Italy; 2010.
33. Taherdoost H, Sahibuddin S, Jalaliyoon N. Perceived Barriers and Benefits of Web Based Services. *2014 Int Conf Comput Sci Comput Intell*. 2014:34-39.
34. Yan-qin Mao, Lu Jin SS. Research and Implementation on Autonomic Integration Technology of Smart Devices Based on DPWS. In: *Advanced Intelligent Computing Theories and Applications*. Vol 9227. ; 2015:179-186.