PERFORMANCE ANALYSIS OF A UPNP/DHCOMPLIANT ROBOTIC ADAPTER FOR COLLABORATIVE TASKS DEVELOPMENT

Alejandro Álvarez Vázquez [1], Ignacio González Alonso[1] and M.P. Almudena García Fuente[1]

[1]Department of Computer Science. University of Oviedo. Oviedo, Spain.
alvarezvalejandro@uniovi.es, gonzalezignacio@uniovi.es, agarciaf@uniovi.es

ABSTRACT

This paper describes the performance analysis of an adapter in accordance with standard UPnP DHCompliant (Digital Home Compliant) for a service robot. The DHCompliant adapter has been developed to solve some limitations that UPnP protocol suffers and to develop new DHC concepts. Moreover, it showcases with a particular example how the open protocol DHC is useful for the development of collaborative tasks, localization, energy management and other fields altogether. That interoperability is being done between devices obtaining a virtual device which can obtain the control point logic and the device logic simultaneously.

KEYWORDS

Device, DHCompliant, control point, UPnP, adapter, collaborative tasks.

1. INTRODUCTION

The main problem analyzed in this paper is part of the question of interoperability between automation systems [1] and in particular, to understand what is the minimum requirements to say that two devices are interoperable. Indeed, it is important to remark that the interoperability among systems requires more than the simple exchange of information. Moreover, companies have to facilitate the use of the same platforms and different languages, and in addition to that, to other important aspects, such as the ability of interaction and the implementation of some collaborative tasks [2], localization, etc.

One of the commercial standards for interoperability is the UPnP protocol [3]. UPnP is a protocol for data transmission which only requires version-compatibility at protocol level, not at programming environment / language / run-time / OS level. Also, UPnP does not require vendors to develop on a specific OS, language, and hardware. There were other standards for the same purpose, but this research had the prerequisite of Zero-Conf for the final user, so they are not going to be analyzed in deeper detail.

An application of this protocol is home automation [4], in which it gives the possibility to create the distribution control architectures. The main idea of UPnP model is the use of two different components; the control point (CP) and the device. So it helps to make every device (a robot, a router, etc.) accessible to the rest of the nodes in a local network (LAN). On top of UPnP, DHCompliant was implemented.

DOI : 10.5121/ijdps.2012.3101

1.1 DHCompliant

DHCompliant project [5] aims to integrate home automation and robotics in the digital home network based on the UPnP [3] technology. The protocol is divided into a number of subsystems that can meet the existing needs in a home automation environment. It is a protocol set up over UPnP and it includes the following subsystems: Groups [6], Localization [7], Intelligence [8], Energy [9], and Security & Privacy [10]. A key part of the DHCompliant protocol is dedicated to make the devices that are connected to the network being able to communicate among themselves (to increase their integration degree). These devices need a software component that allows them to communicate with the protocol, the *DHC adapter*. Then, the DHCompliant architecture and protocols propose a solution to develop collaborative tasks between robots taking into account the information which the home automation devices can provide. For instance sensor information such as lighting conditions, humidity parameters or presence detection, among many others. All the information, from the automatic discovery of devices to remote invocations of robot actions, is handled to perform tasks managed by the UPnP protocol, in the mentioned Zero-Conf context for Service robots (or devices).

1.1 Service robots

A service robot [11] is a robot that operates partially or totally independent, to perform useful services to human welfare and equipment, excluding manufacturing operations [12].

Nowadays, service robots have been used for personal use (entertainment, education ...) and domestic tasks (household chores, cleaning mostly). They form a different market with different distribution channels (department stores, internet) and much lower prices compared with robots for professional use. However, it still represents the largest market for robotics in terms of number of robots, and currently there have been sold over 5 million units (3.3 million vacuum units and 111.000 of mower robot units (lawn mowing robots)), and there are already more than 2 million entertainment robots (toys, mobile humanoid robots to assemble and program, etc...) in homes and schools. On the other hand, the actual number of other home robots, such as surveillance robots (fire detection or strangers in the house), or to help disabled and old people (wheelchair robot for example) are still relatively small [11].

The robot used in this research was the robot Rovio [14], which is a Wi-Fi robot with a webcam that allows you to see, hear, and speak from anywhere. It is easily controlled from a computer or mobile phone which has Internet connection. The built-in web camera has full audio and video streaming and Rovio´s system audio / video is compatible with Skype and MSN Messenger. But the main innovation is its localization system, allowing getting the coordinates of the home.

The objective of this research is to explain the creation of an adapter to provide the skills needed to communicate a robot within a network to other UPnP devices in the framework of the project DHCompliant. The final results will be demonstrated on a real robot.

The main objectives that the adapter must cover are the following:

- Genericity: Is very important to create a generic adapter that provides the common base for the creation of adapters for any robot.
- Extensibility: The proposed system should allow extending its functionality in a simple and clear way.
- Abstraction: The steps to create the adapter must release the manufacturers of tasks related to the understanding of low-level UPnP protocol.
- Open source: Development tools must be preferably open source to facilitate the access to them for manufacturers and possible modifications to suit the needs of each if necessary.

The following sections of the paper describe how to build a DHCompliant adapter, then, the experiments and results are stated, and finally it was discussed the conclusions and future work.

1. BUILDING THE ADAPTER

In this section we explain how the process of building a DHCompliant adapter was. First, what were the tools for doing it, second how was the development process, later, which was the adapter architecture chosen, and finally the adapter activities, as SysML open standard activities and sequence diagrams.

1.1 Tools for creating the adapter

Open Source Developer Tools for UPnP™ Technologies [13] have been used for developing the *DHCompliant* adapter that enables the creation of the skeleton of what will be the full adapter.

- *Service Author*: A tool for defining *UPnP* services which is capable of generating an *XML* file using the actions implemented and the related state variables specified by the *UPnP* service. This generated file is consumed by the *Device Builder* tool.
- *Device Builder*: This tool creates the *UPnP* device from the *XML* files that describe the services. It can be specified through all the device attributes such as the *deviceID, FriendlyName,* or indicate if the adapter represents a device or a control point. After defining the attributes of the device and related services generated with the utility *Service Author,* the tool generates the *UPnP* stack. This generation is the creation of a source code project with the initial skeleton in order to begin to implement the actions of the services and other logical.
- *Device Spy*: Utility to visualize *UPnP* devices connected to the network and its services, actions and state variables. In the following picture it can be seen the *Device Spy* software working.

1.1 Development process

In order to be able to develop all the system, it is necessary to know deeply the robot API. In this case it has studied the Rovio © robot API available in the manufacturer's website [14]. This project was developed using C# language so the RovioLib [15] library for .Net was the best choice to communicate with the robot.

As seen in the tools section, UPnP tools provide the skeleton of the UPnP device, the UPnP stack, which is generated through the combined use of the tools described. Figure 1 shows the steps followed in order to create the adapter. The first step was to define the services, actions and state variables with Service Author, then it was created the device with Device Builder using as input XML files generated in Service Author. At this point, the Device Builder allows us to export the created device to the source files, which will be the starting skeleton mentioned above.

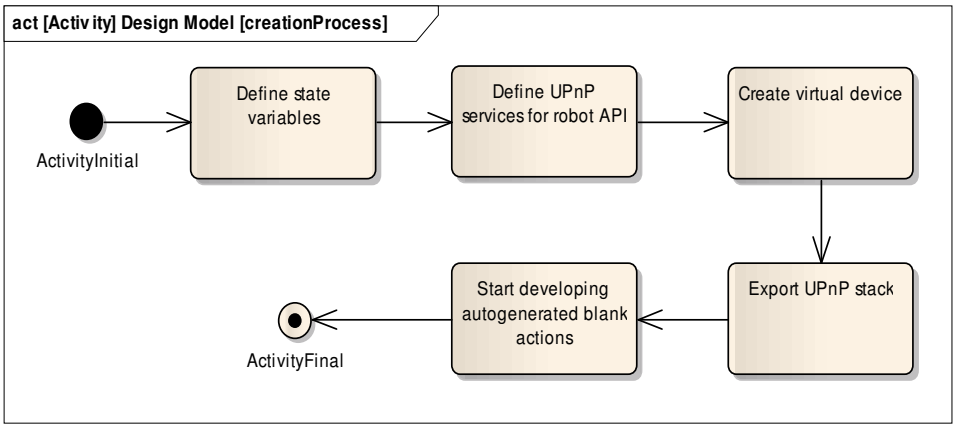

Fig. 1 Activity SysML diagram for creating the adapter

Once it has been obtained the main structure of the adapter, it is necessary to implement all the services that *UPnP* adapter will provide. The adapter must provide the services of *DHCompliant* as well as the own robot services.

1.1 Proposed architecture

The architecture of the proposed solution for the *Rovio* has four main sections, illustrated in figure 2:

- *DHCompliant layer*. All DHC modules are located in this layer. All protocol features are provided by these modules.
- *UPnP layer*. This layer is responsible for the discovery and management of subscriptions in the *UPnP* network.
- *Web server*. It is the link between the *UPnP* network and the physical robot. In this case, the mechanism to send commands to the *Rovio* is made through *CGI* requests.
- *Physical robot*. Is the actual physical device.

All this together forms the virtual device DHCompliant

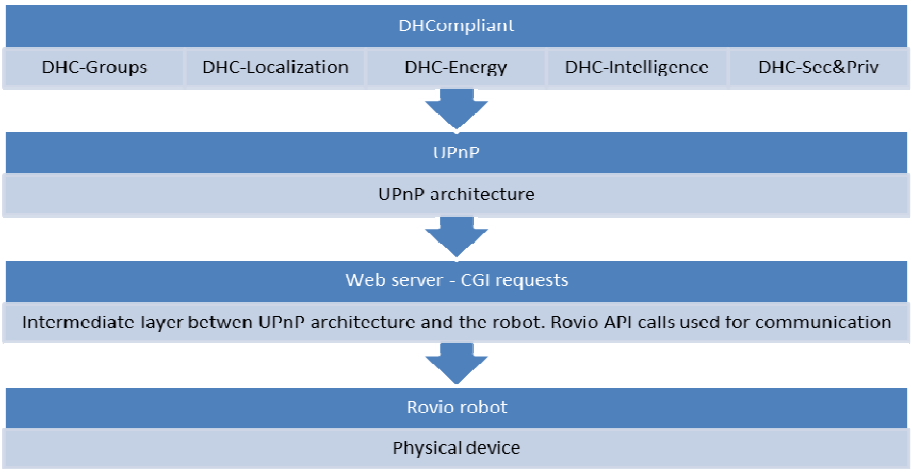

Fig. 2 Adapter architecture

For this experiment it has been taken account several services and actions. The main services that are included in the adapter are the following:

- Camera control service
    - Change resolution
    - Get camera video
    - Get image
- Mail service
- Movement control service
    - Get status
    - Go home
    - Manual drive
    - Email image
- Network service
- Time service
- DHC Groups service
    - Cancel task
    - Get task request
    - Set robot data
- DHC Localization service
    - Get coordinates

One of the problems encountered in the experiment was to find a way to provide the intelligence for the virtual device not only to be able to publish its features and services to the network, but also to be able to find out other devices and manage subscriptions and events alone. For solving this issue, it has been studied in depth the source code of *Open Source Developer Tools* [13] mentioned above. Then, the code was adapted to *DHCompliant* work environment by providing the desired intelligence to the adapter and getting integrated the virtual device and UPnP control point function simultaneously.

1.1 Flow of execution

As seen in the architecture of the solution, three major blocks can be distinguished. The following diagram (Fig. 3) shows the interaction of a generic sequence in which the adapter handles an event in the network.

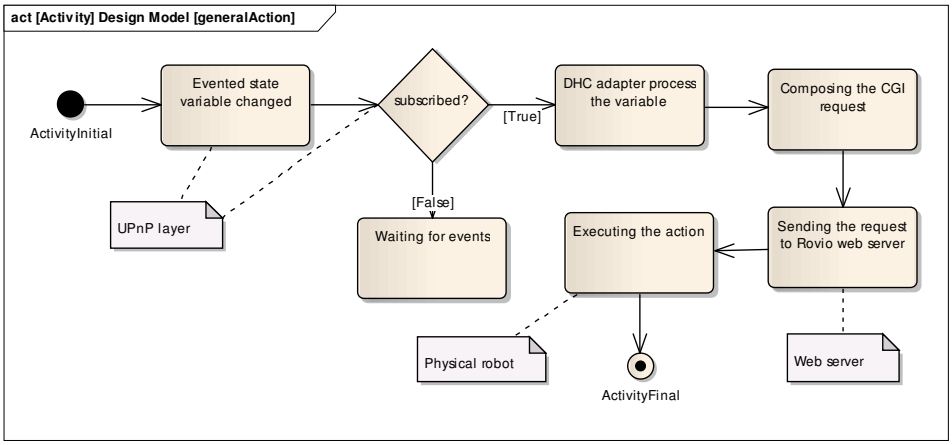

Fig. 3 Activity diagram illustrating the communication between the blocks of the architecture

Firstly, the device is running. Initially it is waiting for events that may occur in the *UPnP* network. Because the adapter contains the logic for an *UPnP* control point, it is able to discover other devices and perform subscriptions to services contained in discovered devices.

Assuming that the *DHC* device is running, and publishing their attributes and services, the adapter recognizes that information and automatically publishes and subscribes to services in *DHC* device.

Once the state variables are modified in *DHC* device, the robot will notice this change and act accordingly. The adapter processes the value of that variable and sends an instruction to the robot. The adapter composes the *CGI* request and then it sends this request to the own web server, which interprets and produces the desired effect on the robot and returns the result of the operation.

In the figure below (Fig. 4), it can be observed the sequence of execution of one robot API instruction, *manualDrive*, which causes movements in the robot. The *s_value* (1-10) parameter specifies the speed and *d_value* (0-18) indicates the type of movement.

AdapterDevice class is the abstraction of the physical robot. This class makes a call to the *UPnP* service related to the robot motion control, *DvMovementControl*. From here, using a *RovioWebClient* object, which is an instance of the robot's web server, the *CGI* request is made and it is interpreted by the *Rovio*. Finally, there is a movement in the robot and the web server returns the result of the operation to be handled by the adapter.

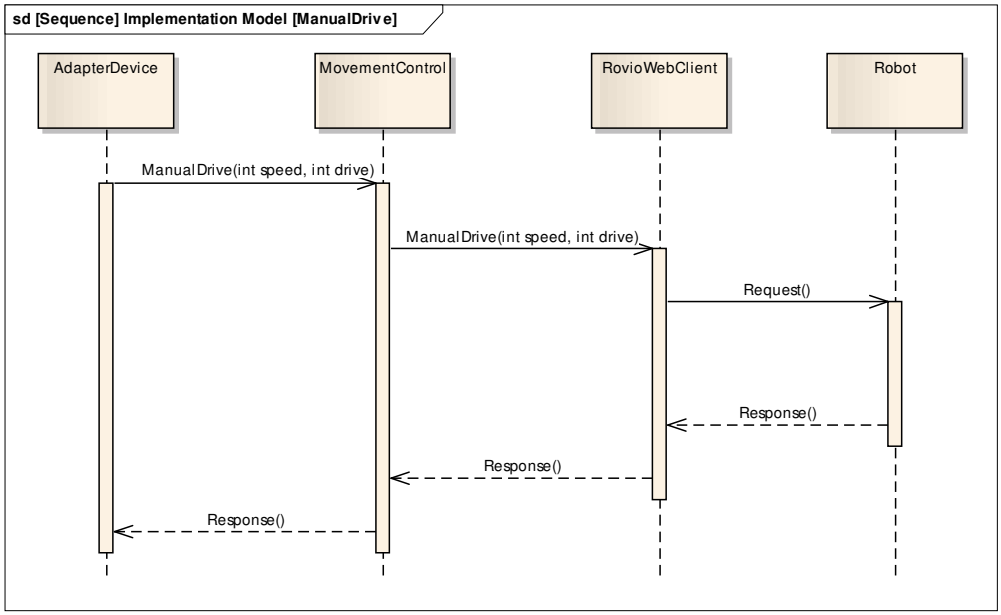

Fig. 4 Diagram showing the sequence of a call to a service robot

2. EXPERIMENTS & RESULTS

In the Experiments & Results section we describe how we set up the network for the experiments, which were the UPnP configuration, services, state variables and messages, and then, how the DHCompliant adapter task was. Finally, we present the performance analysis done.

Name	Value
Base URL	http://192.168.0.25:2024/
Device icon	None
Device URN	urn:schemas-upnp-org:device:DHC-device:1
Embedded devices	0
Expiration timeout	1800
Friendly name	Rovio Wowwee
Has presentation	False
Interface to host	192.168.0.25
Manufacturer	Wowwee
Manufacturer URL	http://www.nuvoton.com/
Model description	Rovio
Model name	Rovio
Model number	2.0
Presentation URL	
Product code	
Proprietary type	
Remote endpoint	192.168.0.25:2024
Serial number	
Services	11
Standard type	
Unique device name	a1c9f8ff-5bd8-4bfa-ace0-d5edb7142399
Version	1.0

Fig. 5 Rovio adapter device details

2.1 Network connection

The experiments were repeated 20 times each. Prior to the DHC adapter experiment, the adapter UPnP features were checked with the following tests using the Device Spy software (Fig. 5). The Table 1 illustrates the results of the UPnP experiments.

Table 1 UPnP network tests

Test	Result OK	Result not OK
UPnP network discovery	16	4
UPnP service publication	16	4
UPnP invocations	16	4
Evented state variables	16	4
UPnP subscriptions	16	4

2.1 UPnP network discovery

With the UPnP network discovery test it was demonstrated that the adapter was able to publish its details to be discovered in the network. At this point, the adapter is running on the UPnP network and ready for communication with DHC. Parameters such as "friendly name" describe the device in the network to be identified by other device and the core modules of DHCompliant. Due to internal problems of the local network of the laboratory where the experiments were performed, in 4 cases it was not possible to discover the device in the network. Therefore, others tests depending on the discovery could not be performed either.

2.1 UPnP Service publication

UPnP service publication and UPnP invocations test the ability of being able to publish its services and actions and the possibility of invoking its actions. The figure below (Fig. 6) shows the services published by the adapter device visible to other DHC devices.

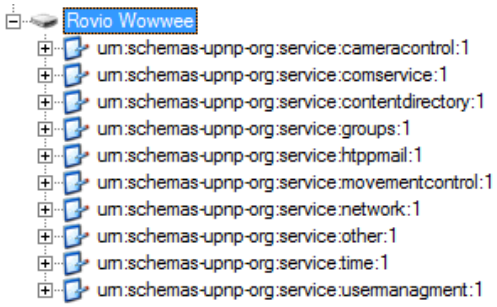

Fig. 6 Some services from the Rovio adapter discovered in the network

2.1 UPnP invocations

The ability to be able to handle UPnP invocations was successfully conducted. Invoking the action ManualDrive from the service MovementControl which was tested and the change reflected in the state variable speed. The screen capture from the device spy software (Fig. 7) shows the feature which allows performing invocations of actions and checks the operation.

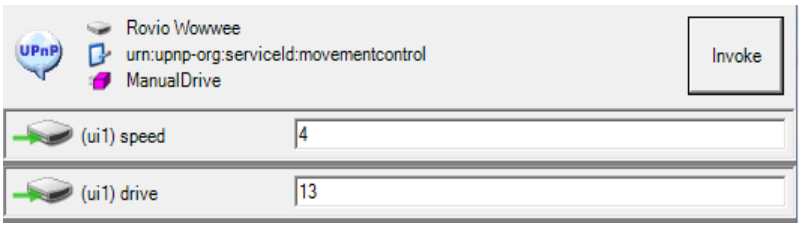

Fig. 7 Action invocation example

2.1 Evented state variables

Evented state variables were checked, too, to know the behavior to a state change. In the figure 8 it is shown the state variables related to the module DHC-Groups.

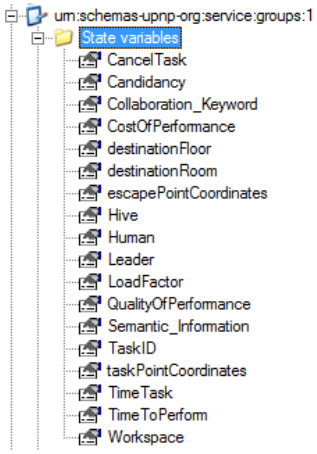

Fig. 8 DHC-Groups state variables

2.1 Task development

An experiment in which DHCompliant is expected to work like the following way has been done. DHC includes the groups and location service. From the user interface a task is created. The parameters of the task are published in the UPnP network and collected by the DHC device that is responsible for communicating with the adapters to find the suitable robots to perform the proposed task. All this process is done through the UPnP message passing between devices. The following table (Table 2) shows the results of the tests done. Apart from the problems caused by congestion due to the internal network, some faults were detected during the experiments. All failures were caused by system implementation errors. For example, a call to a subscription invocation with wrong parameters or handling a state variable with the wrong name causes the system not working properly.

Table 2 Task tests

Test	**Result OK**	**Result not OK**
Discovering *DHC* device	15	5
Subscription to *DHC groups service*	13	7
DHC event handling	18	2
Performing the task	17	3

The following screen capture (Fig. 9) shows that the test results were satisfactory. It can be seen that subscriptions are done and UPnP events were properly handled.

The picture shows a screen capture from the adapter interface. In the status section it is shown the events that have been handled at the moment. There are three events related to register state variables detected in the UPnP network. These are variables from the service Groups from the DHC device. Firstly, three subscriptions were performed, one on each DHC service. Secondly, the adapter handle events related to the state variables and finally with all information given by the services, the robot can perform the task created by the user interface in DHCompliant. The screen capture below shows what is happening through the Rovio's camera.

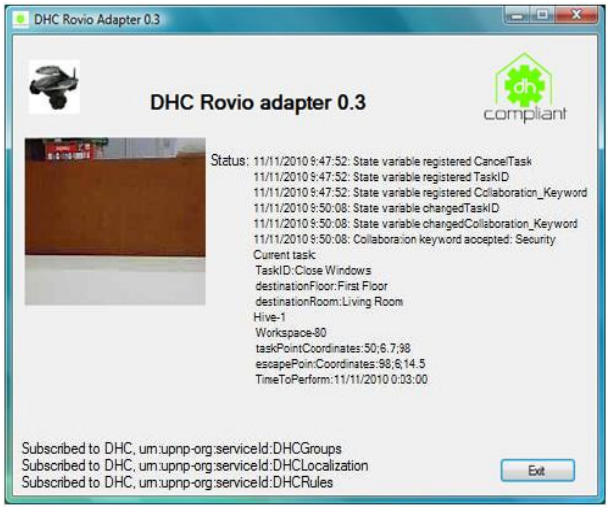

Fig. 9 Rovio adapter screen capture

At the bottom, the interface shows the currently subscriptions. Three subscriptions are done:

- DHCGroups represents the Groups service.
- DHCLocalization represents the Localization service.
- DHCRules represents the rules service.

With this experiment it has been demonstrated the feasibility and operability of the generic adapter created for devices that wish to be compatible with the protocol DHCompliant through a test involving all DHC elements developed so far. The experiment has been repeated several times to demonstrate how efficient is the adapter proposed.

The task created consisted of sending to the Rovio the commands to perform a movement from its charging dock to a specific point to perform a security surveillance task. The course of action is shown in the following capture taken in the Infobótica test arena (Fig. 10). The robot is located at coordinates -149, -89 and must go to a point near the center of the arena (coordinates 0,0). With the navigation system implemented, it is able to calculate the angle, direction and distance to cover. The coordinates are received from a location web service of DHCompliant based on computer vision. If an obstacle is detected by the built-in infrared sensor, the robot is capable of performing a turn and calls back the web service to obtain new coordinates to recalculate the new route.

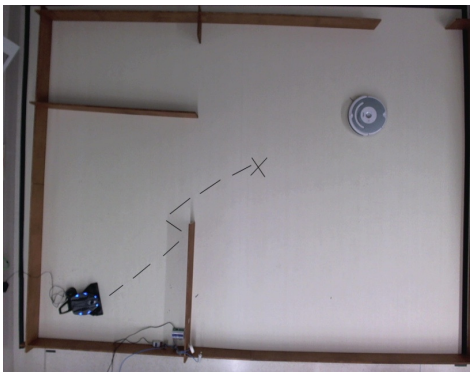

Fig. 10 Infobótica test arena

Other goals proposed have been achieved. The structure of the adapter along with the possibility of its modification and use of the UPnP Tools, give it greater extensibility and make it very easy to add new features. Once the skeleton is created by the UPnP Tools it is not very difficult to add new functions.

It has been obtained a solution which provides manufactures a powerful abstraction mechanism to create and modify the adapter template to suit their needs.

2.1 Performance measures

The previous section explains how the adapter performed as a proof case of the work done, but now we would like to show also how the performance of that adapter in terms of computer resources use was. In particular we analyzed the CPU and RAM consume. The Figure 11 shows the % of time per function consumption in one of the main classes in the adapter. In this case it is shown the AdapterDevice class main methods as a way to explain where the most time consuming activities of the adapter were. It was made with a source code profiler named SlimTune [16].

Additionally, the results of the CPU usage are shown in the Figure 12. The first peak in the graph corresponds to the creation of the virtual device and the UPnP subscriptions handling. The max CPU usage here is 78%. There is a constant usage period in which the adapter is waiting for incoming task requests.

Then, when the adapter processes a task the usage reaches the 55% of total CPU use. The previous data was gathered from the .NET profiler: Your Kit Profiler [17].

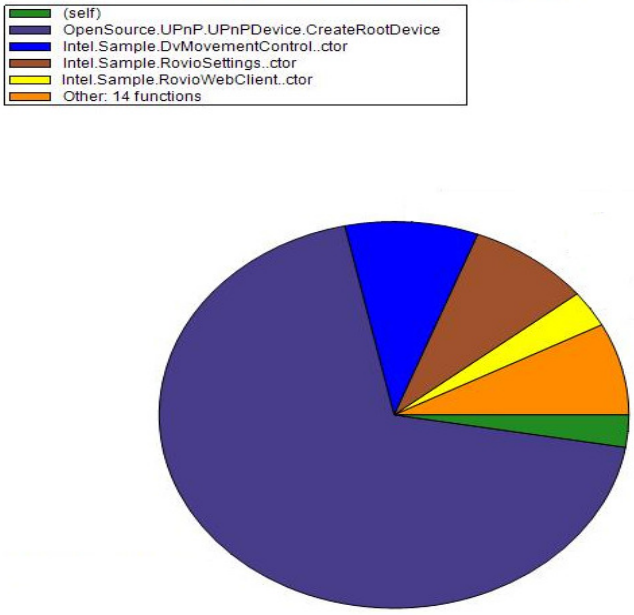

Fig. 11 AdapaterDevice.cs % time per function consumption details

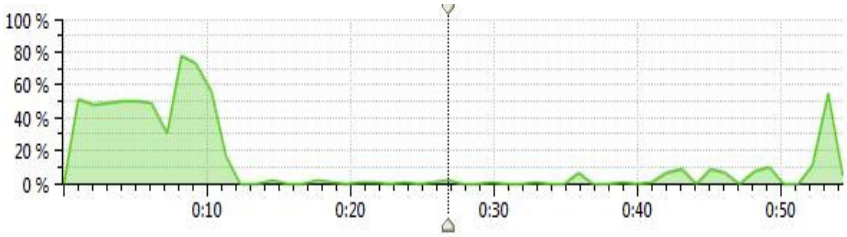

Fig. 12 CPU Usage

Finally, it is shown the memory usage during the launched task (Table 3). As can be seen the adapter is also memory eager software with a 78.21% of memory usage. Moreover, the heap memory (data dynamic memory) use was of 71.87% of the total, so it is also intensive memory use during the execution of the adapter.

Table 3 Memory usage

Memory	**Max**	**Average**
CLR Heap Memory	32 MB	23 MB
Process Memory	101 MB	79 MB

3. DISCUSSION AND CONCLUSIONS

From the previous results we can conclude that the DHC adapter was feasible, and successfully developed but very CPU and Memory eager in two phases of its execution. The Figure 12 showcased the pattern of CPU consuming with high CPU use at the initial connection phase and in the beginning of a new tasks so it can be concluded that new advances to have a better adapter performance should analyze how to reduce that CPU use.

The research results showed how the adapter was developed achieving not only the feature of providing UPnP protocol mechanisms to a robot that did not allow the use of its services through a UPnP network, but also obtain an DHCompliant adapter able to communicate with devices that are part of the DHC protocol.

The proposed adapter satisfies the main objectives of genericity and extensibility, as it provides a common template, generic and easily extensible to create adapters for other robotic devices in a simple and transparent way to the developer. Besides that, it has been achieved an abstraction of lower layers of the UPnP architecture releasing manufacturers from deal with low-level protocol aspects. All of this was achieved with open source and free tools so access and/or modify the adapter source code can easily be made by device manufacturers.

This adapter provides robots the ability to communicate with their home automation environment or other DHCompliant devices, so they can interact with the elements of the digital home and with other robots to execute collaborative tasks. It also provides them the services that DHC is able to offer, for example, the energy service to optimize the energy management and calculate energy costs, the rules service to control the execution of the tasks, the groups service to perform a task in the most optimal way, or the location service to be able to know where the robots are located within the house and to develop tasks where the user needs to be performed.

The obtained results will be used as the basis to establish a common process to develop a generic adapter to be used as the pattern to develop other adapters for state other service robots. This work is included on the of the DH Compliant project (Digital Home Compliant), which is currently developing in order to obtain a more reliable environment, for service robots.

ACKNOWLEDGMENTS

This work was possible thanks to the financial support by the Spanish Department of Science and Technology. We must acknowledge the continuous support given by the University of Oviedo, and also for providing all the resources that made possible our research and, in particular, by their management of the DHCompliant project grant: MITC-09-TSI-020100-2009-359. The DHCompliant consortium is composed by Ingenium S.L., Domotica DaVinci, University of Seville, Movirobotics S.L., Cartif Foundation, ARA (Applied Research Associates) and University of Oviedo.

REFERENCES

[1] Kell A and Colebrook P, «Open Systems for Homes and Buildings: Comparing LonWorks and KNX. i&i limited». [Online]. Available: http://docs6.chomikuj.pl/711810594,PL,0,1,Comparativa-LonWorks-vs-KNX-(ingl%C3%A9s).pdf.

[2] T. Balch and L.E. Parker, Robot teams: from diversity to polymorphism, A K Peters, 2002

[3] UPnP Forum. (2010, Octubre). http://www.upnp.org/. Accesed November 10, 2010, http://www.upnp.org/

[4] S. Dixit and R. Prasad, Technologies for home networking, John Wiley & Sons, 2008.

[5] DHCompliant Project. (2010, Noviembre). DHCompliant. www.dhcompliant.com. Accesed October 25, 2010, http://www.dhcompliant.com/

[6] Infobotica Research Group. (2011). DHC Groups specification. Accesed January 9, 2012, http://156.35.46.38/data/files/Collaborative/DHC-Groups%201.0.pdf

[7] Infobotica Research Group. (2011). DHC Localization specification. Accesed January 9, 2012, http://156.35.46.38/data/files/Localization/DHC-Localization%201.0.pdf

[8] Infobotica Research Group. (2011). DHC Intelligence specification. Accesed January 9, 2012, http://156.35.46.38/data/files/Intelligence/Rules/DHC-Rules%201.0.pdf

[9] Infobotica Research Group. (2011). DHC Energy specification. Accesed January 9, 2012, http://156.35.46.38/data/files/Energy%20Management/DHC-Energy%201.0.pdf

[10] Infobotica Research Group. (2011). DHC Security and Privacy specification. Accesed January 9, 2012, http://156.35.46.38/data/files/SecurityPrivacy/DHC-Security&Privacy%201.0.pdf

[11] I. González, M.F. Ndez, M. Fernández, M.A.D.P.A.G. a, and J. Maestre, Service Robotics Within the Digital Home: Applications and Future Prospects, Springer, 2011

[12] K.Kawamura, R.T. Pack, M. Bishay, and M. Iskarous. (1996). Design philosophy for service robots. Vanderbilt University, Nashville.

[13] Open Software Projects. (2010). Developer Tools - Open Software Projects. Accesed Ocober 25, 2010, http://opentools.homeip.net/dev-tools-for-upnp

[14] Rovio™ - Support : WowWee®. Acessed January 10,2012, http://www.wowwee.com/en/support/rovio

[15] Serg Podtynnyi. (2010). Rovio API Library (.NET). Accesed November 9, 2010, http://roviolib.codeplex.com/wikipage?title=Getting%20Started%20Guide%20to%20Rovio%20.Net%20Devolopment&referringTitle=Home

[16] Slimtune - A free profiling and performance tuning tool for .NET applications - Google Project Hosting." Accesed January 11, http://code.google.com/p/slimtune/

[17] Java Profiler - .NET Profiler - The profilers for Java and .NET professionals. (2010, Diciembre). Java Profiler - .NET Profiler - The profilers for Java and .NET professionals. Recuperado Diciembre 28, 2010, a partir de http://www.yourkit.com/

AUTHORS

Mr. Alejandro Álvarez is a Bachelor Degree in Computer Science specialized on Managment in the University of Oviedo and Informatic Systems Managment Master Technical in the School of "Sagrada Familia de Urgel". He has been working in the Autonomic Code of Asturias research project by the University of Oviedo and developing web applications using Ruby on Rails in "Vorago". Right now he is working in "Infobótica", a research group of his university, on the developing of robotic and domotic software.

Dr. Ignacio González Alonso (M'09) was born in Oviedo, Spain in 1978. This author became a Member (M) of IEEE in 2009. He grades as a computer science engineer (University of Oviedo, Spain 2002); he had also his Master in Informatics engineering (University of Oviedo, Spain 2006); and finally he has been named Ph.D. In Advances of Computer Science: with its thesis about generative programming in May of 2009. He was the co-founder Criptonet (2001-2002) and Negocios y Robótica. since 2005. He is working as an Assistant Professor at the University of Oviedo, Spain. Moreover, he was the first author of the conference contribution named: "Interoperability Standard used by Service Robots", that received the best paper award at the ICONS2010 conference.He is also the author of "Robots in the smart home: a project towards interoperatibility"publication in the Inderscience publishers of distinguished academic, scientific and professional journals. And he is the co-author of the Springer book "Service robotics within the Digital Home: Applications and future Prospects". He is interested in the fields of Interoperability of heterogeneous systems, Model-driven Systems engineering, Human-Robot Interfaces, Environmental technologies &Energy management, among other technologies. Dr. Gonzalez Alonso is associated with SAE International, INCOSE, EUROP/EURON, OMG, EPoSS, AER-ATP, ASIMELEC and HISPAROB tecnological platforms.

Dr. Mª del Pilar Almudena García Fuente was born in Oviedo, Spain in 09/11/1961. She obtained the grade as a mining engineer (University of Oviedo, Spain 1987); and finally she has been named Ph.D. in 1996. She is working as a Professor at the University of Oviedo, (Spain since 1989). She had three books in the publication services of University of Oviedo titled "Introduction to structured programming and Object-Oriented Pascal", "Fortran programming language" and "Introduction to Computers". And she is the co-author of the Springer book "Service robotics within the Digital Home: Applications and future Prospects". She is interested in the fields of Interoperability of heterogeneous systems, Model-driven Systems engineering, Human-Robot Interfaces, Environmental technologies, Energy management, Home&Building automation, agro-forestry and logistic, industrial and manufacturing supportive technologies.