International Journal of Smart Home
Vol.3, No.2, April, 2009

Flexible Smart Home Architecture using Device Profile for Web Services: a Peer-to-Peer Approach

Jorge Parra[1], M. Anwar Hossain[2], Aitor Uribarren[1], Eduardo Jacob[3], and Abdulmotaleb El Saddik[2]
[1]*IKERLAN-IK4 Technology Research Center, Spain*
[2]*Multimedia Communications Research Laboratory, University of Ottawa, Canada*
[3]*DET, Faculty of Engineering, University of the Basque Country, Spain*
JParra@ikerlan.es, anwar@mcrlab.uottawa.ca, AUribarren@ikerlan.es, Eduardo.Jacob@ehu.es, abed@mcrlab.uottawa.ca

Abstract

In this paper we propose the design and development of a flexible smart home architecture using a peer-to-peer (P2P) approach. We specifically focus on two distinct aspects of this proposed architecture. First, we analyze how the different home devices and services can be represented as individual peers in order to have a decentralized system, which is scalable by nature and avoids the single point-of-failure usually attributed to a centralized server. Second, we investigate the distribution of application workflow logic among the peers to develop a flexible home architecture with autonomous behavior of the peers. We analyze the suitability of Devices Profile for Web Services (DPWS) to realize the proposed P2P-like architecture for the smart home. We further show how to distribute the application workflow logic among the peers and yet achieving the same global behavior of the system. Our experimental results show that DPWS provides tools and techniques, in particular its discovery and eventing mechanism, which can be leveraged to provide flexibility and autonomy in the overall architecture.

Keywords: *Flexible Smart Home, Device Profile, Web Service, Smart Home*

1. Introduction

There is a growing demand in designing a flexible smart home architecture that aims to interoperate among heterogeneous sensors, actuators and other services. The objective of developing such architecture is to support the users in the pervasive home environment such that they can access any service in a seamless fashion. A flexible architecture should seamlessly incorporate newly evolved services and/or enable modifications to existing services with minimal effort [1], [2], despite the fact that the new or existing services may be tied to a different vendor with vendor-specific interface. In order to fulfill this goal, several technologies have evolved such as OSGi [3], UPnP [4], Web Service [5], Jini [6], HAVi [7] and so on. These technologies provide the option to interconnect heterogeneous devices and services; however, not all of them can equally accommodate the others under the same hood. One of the very promising technologies that can be used to address this issue is the Devices Profile for Web Services (DPWS) [8], which can work with these technologies and provide interoperability among the different devices and services in a smart home environment. In this paper, we will investigate the suitability of DPWS in developing a flexible smart home architecture.

Besides interoperability, a smart home devices and services should act like peers in order to achieve more flexibility and scalability, where the peers can act autonomously on their behalf [9], [10]. This is also justified by the fact that more and more devices and sensors are equipped with increased networking and computation power, thereby providing the options of adopting a P2P-like infrastructure. However, how to develop a P2P-like infrastructure composed of different devices and services is a challenging issue. In this paper, we explore the suitability of DPWS for realizing a P2P-like infrastructure for smart home.

The possibility of having P2P like infrastructure for a smart home opens the door to investigate the distribution of application workflow logic on different peers. It will not only remove the burden of a centralized entity to maintain the complex application workflow to realize different scenarios, but also simplify the application workflow at a peer level, as each peer will be responsible to perform its own objective task depending on the environment notifications. For example, a lamp service peer may only be responsible to switch-on or switch-off the physical lamp device given the application logic it has. Moreover, having a true P2P-like architecture will enable the addition or removal of a service without affecting the overall operations. However, existing solutions are mostly geared towards a centralized control mechanism where the central entity is solely responsible to command how the individual device and service will react in response to various notifications from different devices and services. We also investigate this issue of distributing the application workflow logic to individual peers and show the flexibility it provides.

Existing research in the context of smart home has mostly focused on interoperability, service discovery and service composition issues. We can roughly classify these works as a) adopting OSGi framework [11-13], [1], b) adopting Web Services mechanism [10], [14 -18], and c) adopting both OSGi and Web Services together [19]. In addition to interoperability, some of these works stated the motivation of a P2P home infrastructure [10], [9]. However, a clear description of the methodology and the distribution of application logic have hardly been addressed in these works.

Our contributions in this paper are two-fold. First, we design and develop a flexible smart home architecture and analyze the suitability of DPWS to interconnect smart home devices and services in a seamless fashion, and show how adopting DPWS we can realize a P2P-like architecture for the smart home. Second, we investigate the distribution of application workflow logic into the distributed peers to achieve simplicity and avoid single of point of failure of a traditional centralized control entity.

The remainder of this paper is organized as follows. In Section 2, we present a motivating scenario in the context of smart home. We comment on some related literature in Section 3, followed by a brief problem description in Section 4. The proposed approach is described in Section 5. This is followed by the implementation details and test as stated in Section 6. Finally, Section 7 concludes the paper with some possible directions for future work.

2. Motivating Scenarios

In this section, we state some motivating scenarios which could represent a usual system behavior in a smart home environment. We assume there are different devices and services such as motion sensors, cameras, pressure sensor in the couch and armchair, light control, HVAC, TV, HiFi, and speakers all connected to a smart home network, as shown in Figure 1. In such a setting, a system performs several tasks to support the user's need. For example:

- When a user enters the room (detected by motion sensor and/or identified by camera), the lights are turned on, and the HVAC is activated.

- When a user sits on the couch in front of a TV, the TV is switched on, light level is dimmed to low and the speakers are switched to TV to output the sound.
- If he/she sits on the armchair close to the window with a book, the lamp close to the armchair is automatically turned on, the ambient music is selected and the speakers are switched to HiFi system with a low volume setting.
- When the user leaves the room, all the devices are off (lamps, HVAC, TV, HiFi).
- At any moment, a GUI in user's phone allows him/her to manually control any of the artifacts in the room.

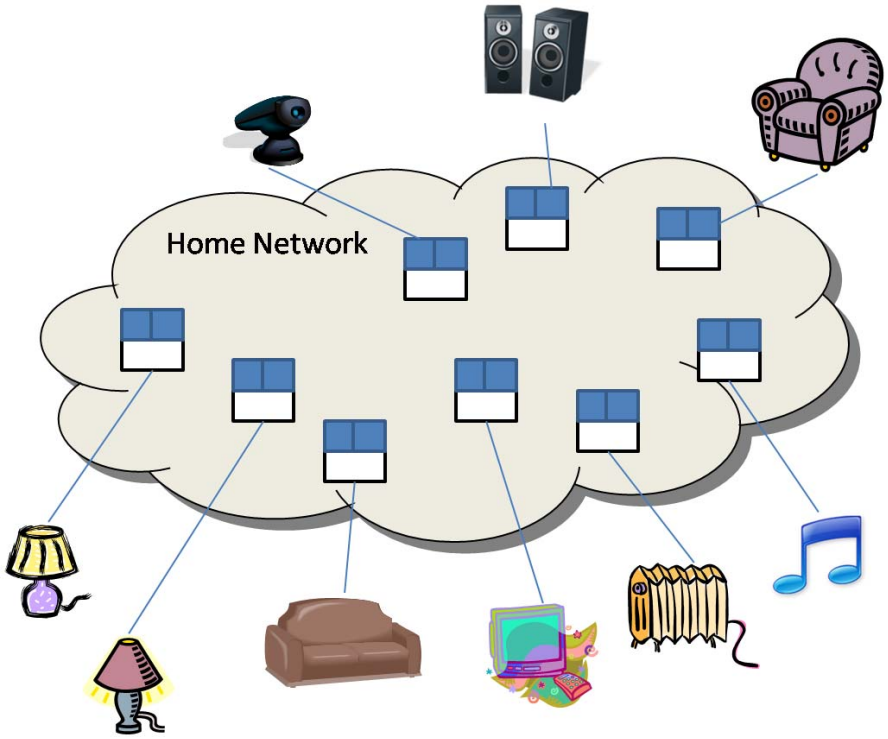

Figure 1. Several devices are connected to a typical smart home architecture

3. Related work

In this section, we comment on some related literature that justifies the use of Service Oriented Architectures (SOA) such as OSGi, Web Service, DPWS and a combination of these technologies. We also comment on some early work that serves as the motivation of conceptualizing the P2P-like architecture for the smart home environment.

The Gator Tech Smart House [11] is an early demonstration of a smart environment, which is based on OSGi framework. This provides a platform-centric approach such that the devices and sensors are all connected to an OSGi gateway, where the application layer acts as a centralized controller to invoke and/or compose different services for the user. However, the P2P-like aspect is not analyzed in this study. Similarly, the works in [1] and [12] propose the use of OSGi as a residential gateway, where the different devices are connected to a central platform using OSGi as a middleware. Although, this approach provides a local solution to expose the devices as services, it requires external mechanisms such as Web Service or UPnP

to intercommunicate among the different OSGi platforms. The peer concept is not explored in these works.

Another branch of work proposes the use of Web Services for developing scalable home architectures [10], and more generic Ambient Intelligence based systems in [15]. Recently, in [17], the authors proposed Web Services as the interoperability mechanism to mediate among heterogeneous technologies, which was also studied in [14] for incorporating legacy home devices into the smart home architecture. In these solutions, Web Services are used solely for communication purpose (SOAP, WSDL and UDDI), without utilizing the facilities offered by other WS family of technologies like WS-Discovery, WS-Eventing, and so on, that have incorporated in the DPWS specification. On the other hand, authors in [16], [18] justify the use of DPWS for service oriented communications to have dynamic Web Service infrastructures, where the devices can be discovered, described, and subscribed using Web Service based standard protocols.

The combination of different technologies like OSGi and DPWS has been adopted in [19]. In this work, the authors used DPWS to integrate external devices in the OSGi platform as local objects and advertise the local objects as DPWS devices, which can be discovered and used by clients external to the OSGi platform. Although the use of DPWS has been explored in this work, the authors have not investigated the P2P-like architecture, where individual peer can autonomously act to execute its logic. This is due to the fact that in this approach, a centralized application composes several local services at the gateway platform.

Unlike the above works, which are based on some central control mechanism, [10] and [20] give an initial motivation to view the smart home architecture as a P2P-like architecture. The authors in [9] proposed a mobile agent-based P2P model for intelligent appliances at home using multiple OSGi platforms, which are interconnected via Web Services. The mobile agents are used to augment the interaction mechanisms among the distributed platform.

In our proposed work, we adopt a fully decentralized P2P model for representing the smart home devices and distribute the application logic to the individual peer so that a peer can take its own decision according to its own rules set by the user.

4. Problem Description

In this paper, we propose the design of a P2P-like smart home architecture and investigate the options to distribute application logic to individual peers. In the following, we briefly describe the problems associated with these issues.

- Essentially home network devices should act like peers [10, 9]. Current SOA-based technology is oriented towards client-server based solutions. For example, UPnP essentially enforces a centralized architecture. It specifies two roles—one is the control point (client) and the other is the UPnP device (which is a service). The control point can discover the individual UPnP devices, subscribe to their events and control them in a pure client-server mode. In this fashion, all the application logic (workflow) resides on the control point thereby jeopardizing the reliability due to the option of single point of failure. Also the central control mechanism can suffer from scalability issue because the number of connected devices directly affects the workload of the control point [20]. Therefore, a P2P-like architecture would make more sense as it provides scalability and extendibility. However designing a P2P-like architecture for the home poses some challenge such as, the definition of roles of the individual peer, management of distributed deployment of the peers and so on.

- Even a P2P-like architecture, from the deployment point of view, may adopt a centralized approach to execute its application logic such that one of the peers can be responsible for orchestrating the actions for rest of the peers. This orchestrator will receive all the message/notifications from the distributed peers and decide based on the application logic what to perform. This approach also suffers from the same problems encountered in a centralized solution.

Our objective is to investigate how to build a P2P-like smart home architecture and how to distribute the application workflow logic to the different peers.

5. Proposed approach

In Section 5.1, we first state the requirements of a flexible smart home architecture and highlight some aspects of DPWS as a technology of choice for our proposed architecture. Next, in Section 5.2., we elaborate the design of the proposed P2P-like smart home architecture, and in Section 5.3 we describe our approach of distributing the application workflow logic among the distributed peers.

5.1. Requirements and DPWS overview

A smart home environment should be an active space where new devices can be incorporated without requiring complex installation or update process (ideally, full plug & play solutions). The desired situation could be described as follows: get a new device, plug it in the home network, and immediately work together with existing devices, performing new individual and collaborative tasks, enhancing the overall functionality, but without a complex setup process.

The requirements that arise from this view can be summarized as follows:

- Mutual discovery of devices and services: existing devices/services should be able to discover the new device, and new device should be able to discover existing infrastructure.
- Mutual interaction: existing devices should be able to benefit from new device functionalities (actions and events), and similarly, the new device should be able to benefit from existing devices/services (actions and events).
- Mutual notification: each device should be able to generate events based on internal trigger conditions and to notify other devices interested in its events, by means of publish-subscribe mechanism.
- Autonomous behavior: every device (existing and new) should know its capabilities and functionalities and should react to external messages received from other devices (direct action invocations or shared events). This could be achieved having some rule based mechanism in the devices. We propose the use of ECA (Event, Condition, and Action) rules for distributed behavior. The application of ECA rules for the smart home has been studied from a different point of view [23], [24]. Thus, any device could respond to external events and according to some conditions should perform some internal actions.

In this work we propose to use DPWS [8] (a set of WS-* protocols) as a solution for fulfilling these requirements. Although the service orientation of DPWS has been thoroughly described in [18], we will briefly introduce some of the features proposed by DPWS specification that are related in our context.

DPWS enables a service architecture built on top of a set of Web Service specifications with two well-defined roles: clients (controlling devices) and services (controlled devices). WS-Discovery, WS-Eventing and WS-MetadataExchange are on top of the protocol stack, allowing the clients to discover, subscribe to events and get descriptions form services using well-known, standard and open protocols as shown in the Figure 2.

Next we briefly describe the WS-Discovery [21] and WS-Eventing [22] protocols as basic elements for the proposed P2P-like architecture.

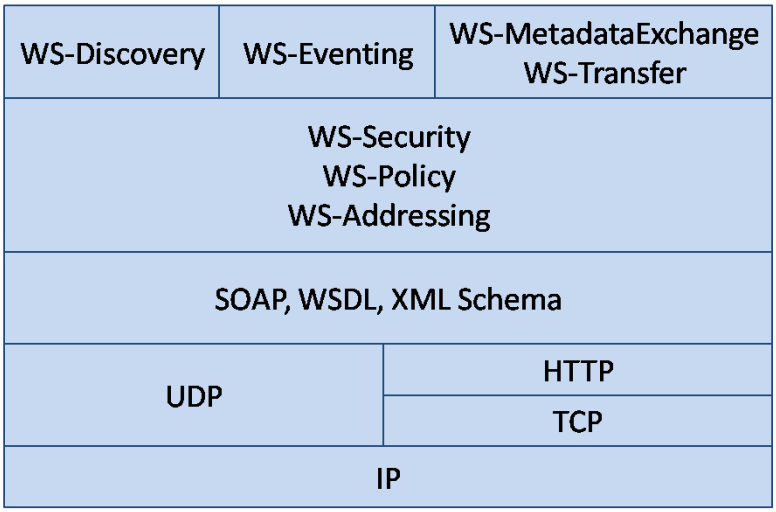

Figure 2. Protocol Stack defined for Devices Profile for Web Services

Table 1. Set of messages defined in WS-Discovery protocol

Message Type	Sender	Description
Probe	Client	Multicast message sent to search Target Services by type or within a scope
ProbeMatch	Service	Unicast response to the sending client when the Target Service matches a Probe message
Resolve	Client	Multicast message sent to search Target Services by name
ResolveMatch	Service	Unicast response to the sending client when the Target Service matches a Resolve message
Hello	Service	Multicast message sent when joining the network containing descriptive information
Bye	Service	Multicast message sent when leaving the network

5.1.1 Web Services Dynamic Discovery (WS-Discovery): This is a multicast discovery protocol defined with the purpose of allowing dynamic discovery and advertisement of Target Services (endpoints that can be discovered) in a network using service scopes and types. Clients (endpoints that search for Target Services) willing to find a specific service can query the network using multicast search messages, and services satisfying the query should reply. These queries can use service name, types or scopes. In order to avoid a polling mechanism in the clients to detect new service availabilities, Target Services also send multicast announcements when joining or leaving the network. In this protocol, no central directory is needed, however to scale to a large number of endpoints, the protocol defines the multicast

suppression behavior subject to the availability of a discovery proxy in the network. It's not intended to internet-scale discovery, but appropriate for home environments, where the number of services is not supposed to be extremely high, and thus, scalability is not a real issue [22]. Table 1 shows the different messages interchanged between a Client and a Target Service in the peer discovery process.

5.1.2. Web Services Eventing (WS-Eventing): It is a Web Service protocol that describes how a client (Subscriber) can register to some events (Subscriptions) of a web service (Event Source). Thus, changes in the service can be notified to any client without requiring standard polling mechanism. The event delivery is accomplished using simple asynchronous messaging. In a typical client-server relation, interactions are always from client to server (e.g. invocation of a service method is accomplished by a message always initiated and sent by a client). Servers are typically passive, and are waiting to serve client requests. On the contrary, the eventing mechanism allows servers to notify clients, and become active. When a change in the server occurs, the server initiates a new communication by sending a message to the clients (Subscribers). To improve robustness, a leasing mechanism is defined so that when an event source accepts a request to create a subscription, it typically does so for a given amount of time. In Table 2, we summarize the messages that are involved in a subscription and notification process.

Table 2. Set of messages defined in WS-Eventing protocol

Message Type	Sender	Description
Subscribe	Subscriber	Sent to an Event Source to create a Subscription
SubscriptionResponse	Event Source	Reply to a Subscribe message sent if the subscription is accepted, stating the expiration date and time of the subscription.
Renew	Subscriber	Sent to an Event Source in order to update the expiration time of a subscription
RenewResponse	Event Source	Reply to a Renew message sent if the subscription renewal is accepted, with the new expiration date and time
GetStatus	Subscriber	Sent to Event Source to request the expiration time of a subscription
GetStatusResponse	Event Source	Reply to a GetStatus message, with the current expiration time of a subscription
Unsubscribe	Subscriber	Sent to an Event Source with the purpose of cancelling a subscription and stop receiving notifications
UnsubscribeResponse	Event Source	Reply to a Unsubscribe message, confirming the subscription cancellation
SubscriptionEnd	Event Source	Sent by an event source when no longer can send notifications to subscribers
Notification	Event Source	Message sent to subscribers with event data

5.2. P2P-like architecture using DPWS

DPWS is based on SOA paradigm and with clearly stated and differentiated client and server roles. However, both roles can simultaneously be implemented in the same component, thus enabling P2P-like architectures [8].

Figure 3 depicts the internal structure of a proposed peer component. We denote P_{Device} to represent the peer component; S_{Device} to represent a port implementing the DPWS Service role in the component; and C_{Device} to represent a port implementing the DPWS Client role. Any device in the home environment can be represented using this model. The business logic uses a device specific API to communicate with the actual physical device (sensors, actuators, artifacts, external services, etc.) and some rules (ECA rules) define its own behavior.

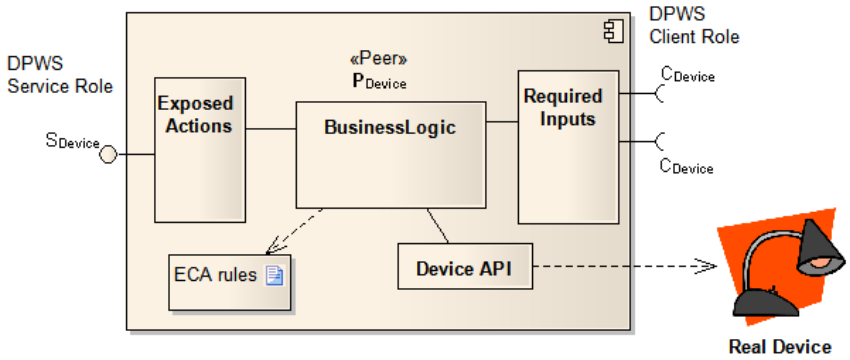

Figure 3. Structure of a peer component

The discovery information, events, description and functional actions of the peer are exposed to the other peers using S_{Device}, leveraging WS-Discovery, WS-Eventing and WS-MetadataExchange, while external inputs needed by the peer can be discovered and subscribed by the peer using C_{Device}. A peer can dynamically connect to as many other peers as necessary, thereby forming a P2P infrastructure as shown in Figure 4, where we depict some of the possible interconnections that could take place in order to realize the proposed scenario. We must also emphasize that any component implementing only one of the described roles could also take part in the network as pure client or pure service peers.

There are different options for a peer to discover other peers in the network. For example, considering that P_{TV} in Figure 4 is interested in locating a specific couch device in the network like P_{Couch}, C_{TV} could send a Probe message with type “Couch” that should be replied by S_{Couch}. In addition, P_{TV} could also discover P_{Couch}, because S_{Couch} advertises its presence when entering the network sending a Hello message that would be received by C_{TV}.

Once P_{Couch} is located, P_{TV} could subscribe to P_{Couch} events. C_{TV} could send a Subscribe message to S_{Couch}, which should reply with a SubscribeResponse message. Then whenever a user sits on the couch, an event will be launched and C_{TV} should receive the corresponding notification message.

5.3. Application logic distribution

In order to explain the proposed application logic distribution approach, we will refer to the smart home scenarios described in Section 2. We will first show how they could be realized using a classical centralized logic approach, where all the behavior rules are centralized in an application component and then explain our proposed logic distribution approach, where the behavior rules are distributed among the peer components. Figure 5 represents the exposed set

of methods and events of the peer components deployed in the P2P based architecture. We will use the same notation described earlier and thus, use P_{Couch}, P_{Heater}, P_{TV}, $P_{Armchair}$, $P_{PresenceDetector}$, $P_{Speakers}$, P_{HiFi} and $P_{LightControl}$ to denote the different peer components. So, the actions and events described in Figure 5 will be exposed by Service role ports of the peers like S_{Couch}, S_{Heater}, S_{TV}, $S_{Armchair}$, $S_{PresenceDetector}$, $S_{Speakers}$, S_{HiFi} and $S_{LightControl}$, and could be accessed by Client role ports of the peers such as C_{Couch}, C_{Heater}, C_{TV}, $C_{Armchair}$, $C_{PresenceDetector}$, $C_{Speakers}$, C_{HiFi} and $C_{LightControl}$.

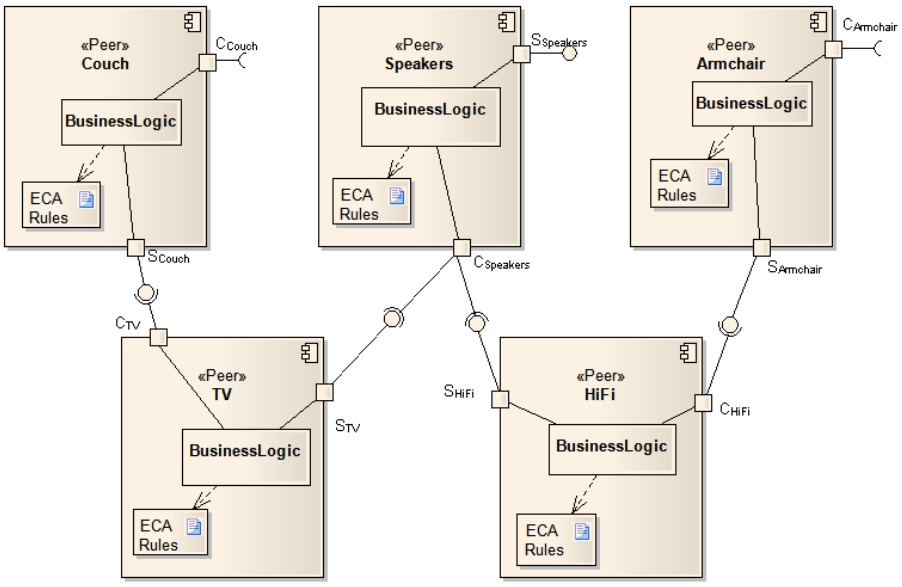

Figure 4. DPWS based P2P connections

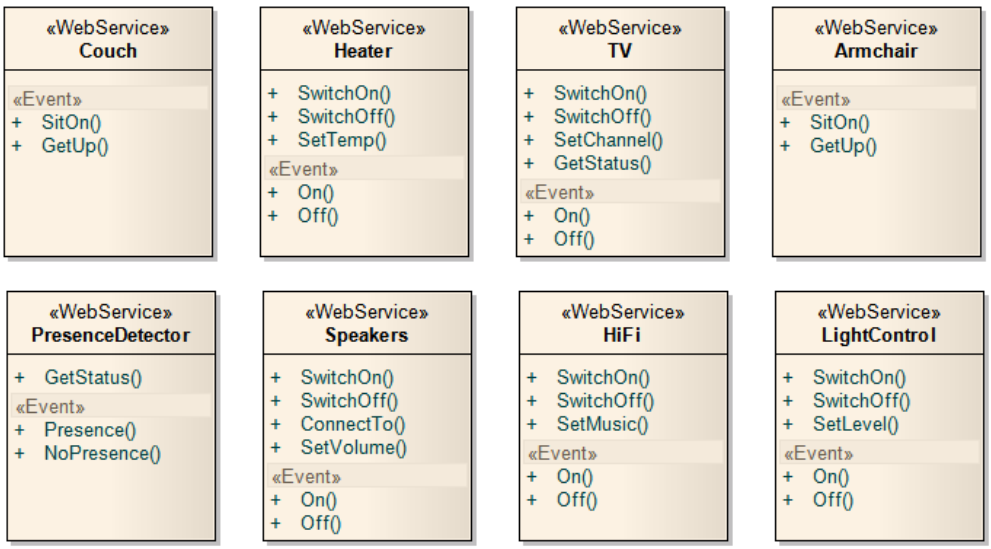

Figure 5. Peers involved in the scenarios with actions and events

By revisiting the overall scenario stated in Section 2, we summarize the following desired system behavior based on which we explain how the application logic can be distributed.

User entering the room:

- Switch on the light
- Switch on the HVAC

User leaving the room:

- Switch off the lights
- Switch off the HVAC
- If TV or music is on, switch them all

User sitting on the couch:

- Reduce the light level
- If the music is on, switch it off
- Switch on TV

User sitting on the armchair:

- Increase light level
- If the TV is on, switch it off
- Switch on the HiFi

5.3.1. Classical approach: using a centralized application logic approach, a client application would be responsible for providing the users with the required system behavior. A pure DPWS client implemented in the application ($C_{Application}$) would be in charge of the different required tasks such as discovering the required services, subscribing to the required events and invoking the required actions of the peers. Figure 6 shows a sequence diagram of one of the described behaviors.

Figure 6 clarifies that when the user initializes the application, $C_{Application}$ sends a Probe message searching for a "Couch" service, which is received by S_{Couch}, which send a ProbeMatch reply to $C_{Application}$. The same process is done with the "TV", "Speakers", and "LightControl" services, which will use S_{TV}, $S_{Speakers}$ and $S_{LightControl}$ respectively to reply with the corresponding ProbeMatch messages. Once all the services have been located, they can be bound to a global application logic workflow that combines all of them in order to perform as expected. As the application must detect the presence or absence of user on the couch, $C_{Application}$ will send a Subscribe message to S_{Couch} in order to receive notifications whenever a user sits on or gets off the couch. When a pressure sensor in the couch detects that a user is sit on it, S_{Couch} sends a notification to $C_{Application}$ that will invoke the SwitchOn and SetChannel methods of S_{TV}, the ConnectTo(TV) method of $S_{Speakers}$, and the SetLevel(LOW) method of $S_{LightControl}$.

In this approach, all the logic resides in the application that orchestrates the actions in the devices, while these are completely passive. If the application fails, all the system behavior is tampered as well.

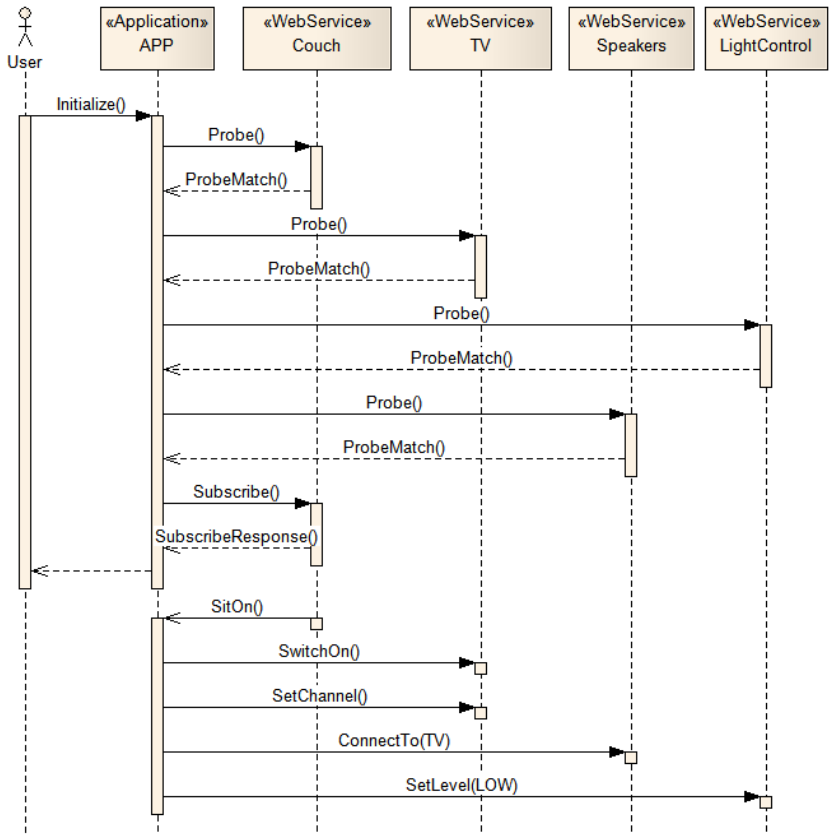

Figure 6. Sequence diagram of a classical centralized logic approach

5.3.2. Proposed decentralized approach: We propose an application logic distribution where devices incorporate a set of rules that can govern their own behavior, following simple ECA rules (as simple as necessary/possible) and thus become reactive: they listen to external messages (notifications coming from other services) and, according to some conditions defined in these rules, they decide to perform their own actions. No centralized application (containing the expected shared behavior) is needed. Each peer is responsible for deciding if its actions are executed or not, according to the defined rules. Figure 7 shows the application logic distribution among peers. The behavior of a peer is described by three well-differentiated sections: Initialization, Event Handling and Operations. In the Initialization Section, each peer subscribes to the events of its interest, while in the Event Handling section, each peer has a set of defined ECA rules that are shown as a programming source code style, but could be implemented using other rule management system. In the Operations section, the peer defines the set of operations that a peer offers.

All the peers are only aware of the events they are interested on, and react autonomously according to the established rules. We must remark that in the collaborative mode, the only messages among peers are event notifications. No action invocation is required, thus reducing the coupling among components. As per Figure 4 and Figure 7, let us consider the peer P_{Couch}, which exposes and allows subscription to the SitOn and GetUp events by means of the service S_{Couch}. Its behavior is extremely simple: whenever the internal pressure sensor detects a user, S_{Couch} sends a notification

message to its subscribers. If we consider peer P_{TV}, we observe that it only listens to the events coming from P_{Couch} and hence during its initialization phase, it will use C_{TV} to send a Subscribe message to S_{Couch} in order to receive SitOn and GetUp notifications.

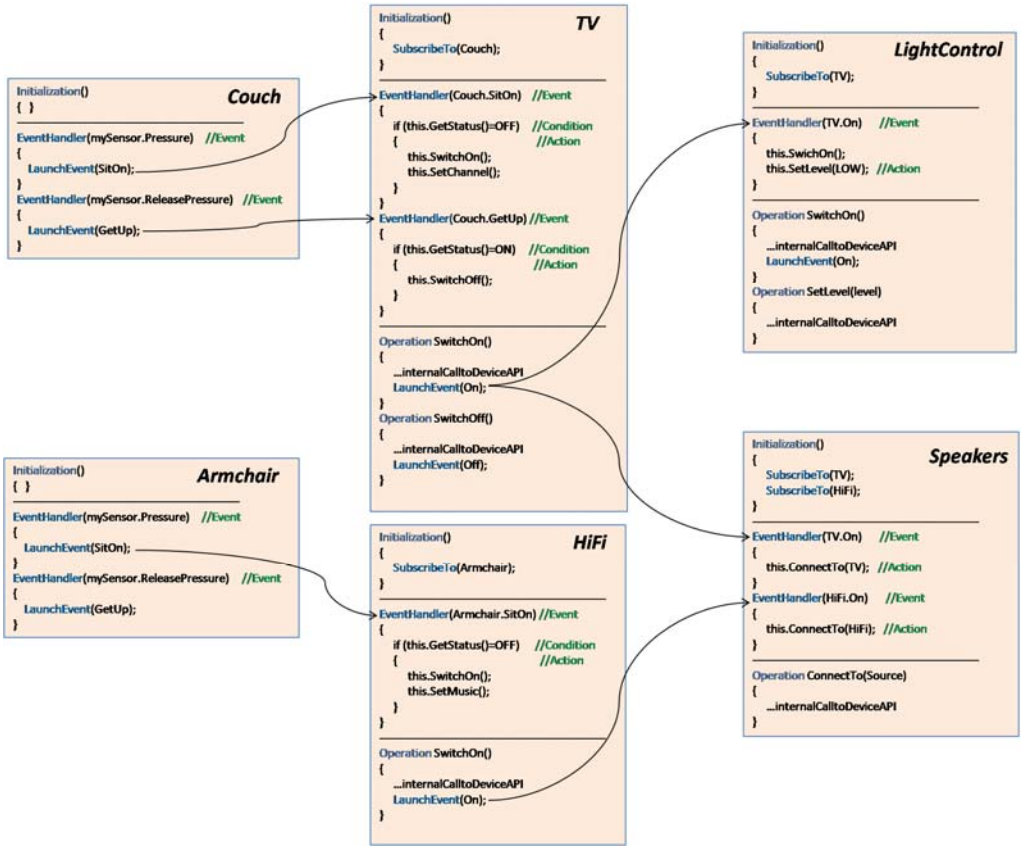

Figure 7. Application logic distribution among peers

When a SitOn notification arrives from S_{Couch}, P_{TV} will check its own status (own action), will switch itself on (own action) and will set the TV channel (own action) reacting to the presence of a user in the couch as expected in the scenario. Further, as a consequence of the invocation of its SwitchOn action, it will launch an event announcing that its status has changed to ON. Thus, any peer interested in this status change will be able to react according to its own behavior. In the proposed scenario, both $P_{LightControl}$ and $P_{Speakers}$ are interested in this event, so both of them will subscribe to it during their initialization phase and $P_{LightControl}$ will react switching itself on (own action) and dimming the light level (own action) and launching an event announcing its new state; and $P_{Speakers}$ will be able to react setting its source to TV (own action). Therefore, the expected system behavior has been achieved distributing the application logic among all the peers, reducing the inter-peer coupling because only event messages are shared.

6. Implementation and test

6.1. Implementation

In order to evaluate the proposed approach we implemented a proof-of–concept prototype architecture based on a set of peer devices with internal ECA rules defined to fulfill the proposed scenario.

Figure 8 shows the implementation environment and how the different devices are connected. We installed Phidget [26] pressure sensors in a couch and in an armchair and connected them using OSGi running in a laptop. We used X10 lamp modules and a PC interface for lighting control connected to the same laptop. We simulated a TV, HiFi system and Speakers services using VLC [27], a media player for various formats with a HTTP interface that allowed us to wrap a DPWS device around it. A GUI in a SmartPhone allowed us to send user actions to the devices.

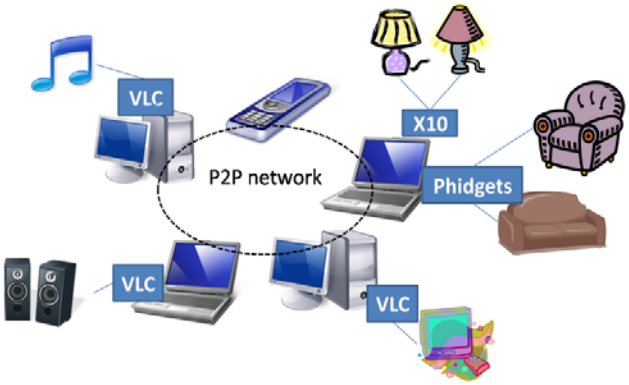

Figure 8. Implementation environment

We used .NET Deployment Framework and OSGi Deployment Framework tools developed in the IST Amigo Project [28] to implement the WS-Discovery and WS-Eventing protocols supporting peers, and distributed them as shown in Figure 8. We remark that some of the peers were implemented using OSGi (P_{Couch} and $P_{Armchair}$), while the others were implemented using .NET Framework. Thus, we demonstrate the interoperability capabilities of a Web Service based solution like DPWS. For this first prototype implementation we hardcoded the ECA rules for individual peers. Figure 9 shows the code snippet for an example LightControl peer. We plan in the future to use an external rule definition representation and a rule parser in the peers in order to allow dynamic update of behavior of the peers, instead of hard coding the rules.

6.2. Test

Our objective was to build a flexible P2P-like home architecture with distributed behavior, so we conducted some test cases in order to demonstrate it.

First, we analyzed the case of incorporating a new device in the environment. To do that, we defined a new scenario to be added to the current ones consisting on automatic TV or HiFi volume decreasing when a phone call is received. In order to realize this scenario, and still support all the others, we incorporate a new peer in the smart phone P_{Phone}, which launches an event whenever a phone call is received. This is all the required behavior for the new peer. We only need to add a new behavior to $P_{Speakers}$, stating that now it should also subscribe to phone events, and define the corresponding event handling actions: when receiving Phone.Call event, set the volume to low level (own action). Thus a simple update in one of the peers supports the newly added scenario.

Second, we tested the case of direct user actions in the environment. We implemented a GUI application in the user's smart phone, which is able to discover and locate all the peer services using a DPWS client (C_{GUI}). Thus, if user wanted to watch TV without sitting on the couch, he/she could use the phone to invoke the SwitchOn action exposed by S_{TV}. This invocation would directly switch the TV on, and launch the corresponding TV.On event that would be received by $C_{LightControl}$ and $C_{Speakers}$, and these two peers would act according to the defined behavior. The light level should be reduced and the Speakers source should be set to play the TV sound in this case.

Finally, we analyzed the system behavior by intentionally removing P_{Couch} from the environment. This change will only affect the P_{TV} peer (see Figure 7). By means of the SubscriptionEnd message (according to WS-Eventing protocol) and Bye message (according to WS-Discovery protocol) sent by S_{Couch}, the unavailability of P_{Couch} was notified to P_{TV}. Furthermore, as a clear benefit of the behavior distribution, user could manually switch TV on, and as showed in previous test case, lights and speakers would still work as expected.

```
Public Class LightControl
    Public Event Light_On()
    Public Event Light_Off()
    Private TV As DPWS

    #Region "Initialization"
        Sub Initialize()
            AddHandler TVService.On, AddressOf TVOn_Eventhandler
        End Sub
    #End Region

    #Region "EventHandlers"
        Private Sub TVOn_Eventhandler()
            Me.SwitchOn()
            Me.SetLevel("LOW")
        End Sub
    #End Region

    #Region "Operations"
        Public Sub SwitchOn()
            X10Lib.SwitchOnLampModule("A3")
            RaiseEvent Light_On()
        End Sub

        Public Sub SwitchOff()
            X10Lib.SwitchOffLampModule("A3")
            RaiseEvent Light_Off()
        End Sub

        Public Sub SetLevel(ByVal level As String)
            ...
            X10Lib.DimLampModule("A3")
            ...
        End Sub
    #End Region
End Class
```

Figure 9. Code snippet showing the behavior of $P_{LightControl}$

7. Conclusions and future work

We have proposed a P2P-like scheme for a flexible smart home architecture and analyzed the possibility of distributing the required application workflow logic to individual peers. In this approach, each device or service represents a peer and acts autonomously based on the application logic it has for its own interaction. We observed that the tools and techniques provided by DPWS, more specifically its discovery and eventing mechanisms, are suitable to realize such a proposed architecture. The proposed approach not only provides the flexibility of adding or removing new or existing devices to/from the home network, it also ensures scalability and removes the burden from the central entity usually encountered in a traditional server or gateway based solutions.

There are scopes for further work in this direction. For example, a) it would be interesting to utilize a rule description language for ECA-rule deployment in different peers, b) the

integration of sophisticated context management framework [25] into the smart home architecture would be a good extension of the current work, and c) to provide users with flexible GUI in order to easily construct the distributed application logic for the individual peers and to perform an extensive usability study.

References

[1] J. Bourcier, A. Chazalet, M. Desertot, C. Escoffier, and C. Marin, "A Dynamic-SOA Home Control Gateway", IEEE International Conference on Services Computing (SCC '06), 2006, pp. 463-470.

[2] Y. Tajika, T. Saito, K. Teramoto, N. Oosaka, and M. Isshiki, "Networked home appliance system using Bluetooth technology integrating appliance control/monitoring with Internet service", IEEE Transactions on Consumer Electronics, vol. 49, no. 4, Nov. 2003, pp. 1043-1048.

[3] OSGi – The Dynamic Module System for java, http://www.osgi.org

[4] UPnP, http://www.upnp.org

[5] Web Service, http://www.w3.org/2002/ws

[6] Jini, http://www.jini.org

[7] HAVi, http://www.havi.org

[8] Devices Profile for Web Services, http://schemas.xmlsoap.org/ws/2006/02/devprof/

[9] C-L. Wu, C-F. Liao, L-C. Fu, "Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology", IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 37, no. 2, March 2007, pp.193-205.

[10] M.Aiello, "The Role of Web Services at Home", Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006), Guadeloupe, 2006.

[11] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The Gator Tech Smart House: A Programmable Pervasive Space", IEEE Computer, vol. 38, no. 3, Mar. 2005, pp. 50-60.

[12] D. Valtchev and I. Frankov, "Service gateway architecture for a smart home", IEEE Communications Magazine, vol. 40, no. 4, Apr. 2002, pp. 126-132.

[13] A. Bottaro, A. Gerodolle, P. Lalanda, "Pervasive Service Composition in the Home Network", 21st International Conference on Advanced Information Networking and Applications (AINA'07), May 2007, pp. 596-603.

[14] M.Nakamura, A.Tanaka, H.Igaki, H.Tamada and K.Matsumoto, "Adapting Legacy Home Appliances to Home Network Systems Using Web Services", IEEE International Conference on Web Services (ICWS'06), Salt Lake City, 2006.

[15]V. Issarny, D. Sacchetti, F. Tartanoglu, F. Sailhan, R. Chibout, N. Levy, and A. Talamona, "Developing Ambient Intelligence Systems: A Solution based on Web Services", Automated Software Engineering, Springer Netherlands, vol. 12, no. 1, Jan. 2005, pp. 101-137.

[16] E. Zeeb, A. Bobek, H. Bonn, F. Golatowski, "Lessons learned from implementing the Devices Profile for Web Services", Digital EcoSystems and Technologies Conference (DEST '07), Inaugural IEEE-IES, Feb. 2007, pp. 229-232.

[17] T. Perumal, A.R. Ramli, C. Y. Leong, S. Mansor, and K. Samsudin, "Interoperability for Smart Home Environment Using Web Services", International Journal of Smart Home, vol. 2, no. 4, Oct. 2008, pp. 1-16.

[18] F. Jammes, A. Mensch, H. Smit, "Service-Oriented Device Communications Using the Devices Profile for Web services", 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07), vol. 1, no. 1, 2007, pp. 947-955.

[19] A. Bottaro, E. Simon, S. Seyvoz, and A. Gerodolle, "Dynamic Web Services on a Home Service Platform", 22nd International Conference on Advanced Information Networking and Applications (AINA '08), Mar. 2008, pp. 378-385.

[20] M. Nakamura, H. Igaki, H. Tamada, and K. Matsumoto, "Implementing integrated services of networked home appliances using service oriented architecture", In Proceedings of the 2nd international Conference on Service Oriented Computing (ICSOC '04), New York, USA, Nov. 2004, pp. 269-278.

[21] Web Services Dynamic Discovery, http://specs.xmlsoap.org/ws/2005/04/discovery/

[22] Web Services Eventing, http://www.w3.org/Submission/WS-Eventing/

[23] J.C.Augusto and C.D.Nugent, "The Use of Temporal Reasoning and Management of Complex events in Smart Homes", Proceedings of European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, August 2004. IOS Press, 2004, pp. 778-782.

[24] Y.Qiao, K.Zhong, H.Wang, X.Li, "Developing event-condition-action rules in real-time active database", In Proceedings of the 2007 ACM Symposium on Applied Computing (SAC '07), Seoul, Korea, March 2007, pp. 511-516.

[25] T. Gu, H. K. Pung, and D. Zhang, "Towards an OSGi-Based Infrastructure for Context-Aware Applications in Smart Homes", IEEE Pervasive Computing, vol. 3, no. 4, 2004, pp. 66-74.

[26] Phidgets – products for USB sensing and control, http://www.phidgets.com

[27] VLC media player, http://www.videolan.org/vlc

[28] IST Amigo project, http://www.hitech-projects.com/euprojects/amigo/

Authors

Jorge Parra received his BS and MS degrees in Electrical Engineering in 1998 from the University of the Basque Country (UPV/EHU), Spain, where currently he is also pursuing his Ph.D. degree in the Department of Electronics and Telecommunications. He has been working in Ikerlan-IK4 Technological Research Centre since 1999 as a full-time researcher in the Software Technologies Area, where he has taken part in a number of projects related to home systems. His current research interests include distributed systems, service oriented architectures and service technologies for Ambient Intelligence systems.

M. Anwar Hossain received the B.Eng. degree in Computer Science and Engineering from Khulna Universty, Bangladesh, and the M.C.S. degree in Computer Science from the University of Ottawa, Ottawa, ON, Canada, in 2005. He is currently working toward the Ph.D. degree with the Multimedia Communications Research Laboratory (MCRLab), School of Information Technology and Engineering, University of Ottawa. He worked for a few years in industry before he completed the M.C.S. degree. His research interests include human-computer and human-environment interaction, information visualization, multi-sensor systems, context-awareness and ambient intelligence. He has authored and co-authored more than 25 publications including refereed journals, conference papers, and book chapters.

Aitor Uribarren received his BS degree in 1998 at Department of Computer and Communication Systems Engineering, Mondragon Unibertsitatea. He completed his MS degree in Distributed Systems and currently pursuing Ph.D. in Smart Home Ambient Personalization at Universidad de Deusto. Since 2000, he's a full-time researcher at Ikerlan-IK4 Research Centre in the department of Software technologies. His main interests are smart home systems, middleware technologies, reconfiguration and embedded system design.

Eduardo Jacob works as assistant professor in the University of the Basque Country (Spain) where he received his BS and MS degrees in Electrical Engineering, in 1987 and Ph.D. degree from the same University in 2001. His work takes place in the Department of Electronics and Telecommunications at Faculty of Engineering of Bilbao where he teaches Operating Systems Laboratory, Mobile Services and Networks and Doctorate Courses on Advanced Networks and Security in Wireless Networks. He has directed several R&D publicly funded projects in security and distributed systems. He is co-founder of the I2T (Research and Engineering in Telematics) Research Group of my University that is currently involved in several projects related security in distributed systems and next generation networks. He is member of the advisory committee in Information Systems of the University of the Basque Country and has been appointed as Technical Expert at the advisory Council of the Basque Agency for Data Protection.

Abdulmotaleb El Saddik (F'IEEE-09) is University Research Chair and Professor, SITE, University of Ottawa and recipient of the Professional of the Year Award (2008), the Friedrich Wilhelm-Bessel Research Award from Germany's Alexander von Humboldt Foundation (2007) the Premier's Research Excellence Award (PREA 2004), and the National Capital Institute of Telecommunications (NCIT) New Professorship Incentive Award (2004). He is the director of the Multimedia Communications Research Laboratory (MCRLab). He is a Theme co-Leader in the LORNET NSERC Research Network. He is Associate Editor of the ACM Transactions on Multimedia Computing, Communications and Applications (ACM TOMCCAP) and IEEE Transactions on Computational Intelligence and AI in Games (IEEE TCIAIG) and Guest Editor for several IEEE Transactions and Journals. Dr. El Saddik has been serving on several technical program committees of numerous IEEE and ACM events. He has been the General Chair and/or Technical Program Chair of more than 20 international conferences, symposia and workshops on collaborative hapto-audio-visual environments, multimedia communications and instrumentation and measurement. He was the general co-chair of ACM MM 2008. He is leading researcher in haptics, service-oriented architectures, collaborative environments and ambient interactive media and communications. He has authored and co-authored two books and more than 200 publications. He has received research grants and contracts totaling more than $10 million and has supervised more than 90 researchers. His research has been selected for the BEST Paper Award three times. Dr. El Saddik is an IEEE Distinguished Lecturer, Senior Member of the ACM, and Fellow of the IEEE.