PERSEO: a system to Personalize the Environment Response through Smart phonEs and Objects

Luca Bergesio, Íñigo Marquínez, Ana M. Bernardos, Juan A. Besada, José R. Casar
Telecommunications School
Universidad Politécnica de Madrid
Madrid, Spain
{luca.bergesio, imarquinez, abernardos, besada, jramon}@grpss.ssr.upm.es

Abstract—**When designing services to control the behavior of a smart environment, it is feasible to rely on mobile devices as mediators to empower the user and handle his needs and preferences. In this direction, this paper describes a concept for mobile-object instrumented interaction through a system that enables the orchestration of a set of smart objects with sensing and/or media capabilities deployed in a room. These objects are coordinated to respond accordingly to the physical and logical state of a mobile device. A specific case of use is described; it is centered on a 'smart night table' that coordinates the response of the smart environment, populated with interfaces such as a TV, tablet, sound system and photo framework. When a user places his smartphone on the smart table, a service checks the active applications in the mobile device and makes an external controller to forward actions to the mentioned interfaces, managing their availability. The prototype shows the viability of these mobile-object instrumented personalization and interaction concepts.**

Keywords-interaction; smart objects; mobile devices; reasoning

I. INTRODUCTION

Nowadays, the most common user devices to ubiquitously interact with digital information or with items connected to the Internet are smartphones and tablets. Thus, even if wearable devices, such as glasses or textiles, are to be the user interfaces of the future, mobile devices with high capabilities in terms of memory, computation and autonomy are today's option to easily handle digital information or to deal with items connected to the Internet.

The concept of smart space is becoming popular to describe an environment equipped with smart objects, which may be orchestrated to satisfy its inhabitant needs. The concept of 'smart object', which may have different levels of intelligence, may be defined as a conglomerate of sensors, actuators, and communication capabilities. In this paper, we will use it to refer to *'a computationally augmented tangible object with an established purpose that is aware of its operational situations and capable of providing supplementary services without compromising its original appearance and interaction metaphor'* [1].

How to personalize the behavior of the smart space by making easy the configuration of smart objects through mobile device is the leitmotif of this contribution. The paper describes a mobile-instrumented interaction concept, which facilitates the coordination between the mobile device's content and actions, and the smart objects in the environment, facilitating the use of enhanced interfaces to visualize and manipulate media data, services and applications.

Section II frames the current research in the state-of-the-art of interaction concepts with mobile devices as enablers. Section III details our proposal, describing a validation scenario for deployment. Section IV gives details about the media management infrastructure. Section V defines the architecture components and the workflow to complete the development. Finally, Section VI shows our conclusions and further work.

II. RELATED WORK IN MOBILE-INSTRUMENTED INTERACTION

Mobile personal devices [2] may have a very important role to facilitate interaction in smart spaces. Touching and pointing paradigms for physical interaction, together with scanning and user-centered object interaction [3] enables simplifying the way we interact with the *smart space.* For example, Want et al. [4] proposed in 1999 a method to link objects to digital resources through RFID tags. Biegl [5] implemented the point&click concept, by using a stand-alone 'remote control' that was capable of obtaining the control information for other devices, allowing operational interaction with the help of a simple user interface.

Among the previous works that have proposed technical solutions to interact with smart objects through personal devices, [5] uses a laser-equipped device, which provides the user with information about the object's control commands by using a small display. When the object detects the laser beam, it transfers the control description to the device by using infrared.

A framework for mobile devices to interact with NFC tagged physical objects that are associated with web services and provide information for their invocation is described in [6]. In [2], smart objects use formatted SMS containing simple commands to enable their remote operation through a mobile device. In [7], it is described a system to easily configure from the mobile phone a chain of event-condition-action modules that can be executed by a smart object.

III. Interaction concept

A. Interaction method

Let us imagine the following situation: a user is inside a room where are available many smart objects with media capabilities, such as a television, a tablet and a hi-fi system. When the user puts his *smartphone* next to/on/beyond a *'trigger' smart object* (not changing the interaction metaphor of this device), the service implementing the proposed interaction method is initiated. The contact/proximity event triggers the automated checking of the applications running foreground in the mobile device, and initiates the consequent action, taking into account: a) the configuration that the user has accepted by default or previously configured (e.g. by using the concept in [7]) for the service and b) his context (e.g. if there are more than one person in the room). For example, if the user is watching photos, the content is transferred to a more convenient interface, in case he is alone or wants to share with the rest of people in the room.

In brief, the interaction method that is proposed in this paper is based on a usable mobile-instrumented way of interfacing and sharing in-device content or application-related actions with smart objects. Section V describes the entities involved in the method, but at this stage of the paper, we can summarize the following basic ones: 1) the smartphone; 2) the trigger smart object; 3) the responsive smart objects, deployed in the smart space; 4) the smart space itself, which is aware of the status of the smart objects when needed.

The driving features for the interaction method are:

- *Mobile-instrumented concept*: the smartphone serves as trigger and orchestrator for the interaction between itself and the smart objects in the environment.
- *Coordinated response mobile device-smart object*: smart objects will respond to the logical state of the mobile device, performing different actions depending on the active applications in it or on its configuration (e.g. silence mode). E.g. smart objects will perform differently if the 'weather' application is active than if there is an incoming call.
- *Physical-based initiation/control without changing the traditional interaction metaphor*: the coordination and control tasks will start when the user interacts with a smart object, trying to preserve the normal interaction metaphor of this object. Additionally, different control actions related to the traditional interaction metaphor of the trigger smart object may be included in the method. Preserving the conceptual models that a user has to interact with an object facilitates learnability.
- *Interface with heterogeneous smart objects*: the concept makes possible to deal with different kind of smart objects. In particular, Section III.a explains a case in which television sets, photo frameworks, hi-fi systems, tablets and computers handle media content from the mobile device, but the proposal is not limited to this and could be extended to other kind of smart objects. For example, if smart metering application to monitor energy consumption at home is active, the air conditioning-heating objects could be responsive.
- *Device-hosted content sharing*: the method enables facilitating environment personalization through the use of media content available in the device. This approach makes sense as smartphones are full of digital content that the user may be willing to enjoy through different smart objects for visualization, mainly to facilitate consumption due to bigger screens or best quality of sound.
- *Rule-based reasoning*: the interaction method is based on a reasoning paradigm based on handling event detection, condition checking and action triggering combinations through rules. In our system, conditions are derived of the applications running in foreground, while actions are related to digital content and application handling (e.g. transferring content from the mobile device to a different object for visualization). Action depend on the foreground applications in the smartphone: for example if the user is watching a video, the action will be to transfer the video to the television; if the user is listening to a song, the action will be to play that song using the hi-fi system. This simple way of handling the logic of the system makes possible to enable easy-to-use mechanisms to personalize and configure behaviors from the user point of view.
- *Context-responsive*: by using the event-condition-action paradigm, the method enables the inclusion of context analysis, in order to adapt the response mobile device-smart object.
- *Flexible architecture*: as it is explained in Section V, the method enables a flexible implementation of its components, which may be deployed in smart objects, mobile devices and infrastructure.

B. Case of Use

To demonstrate the viability of our interaction method, a case of use is proposed. We consider a user with a smartphone inside a room equipped with a television, a photo frame, a tablet, a 'smart night table' and a hi-fi system (Figure 1).

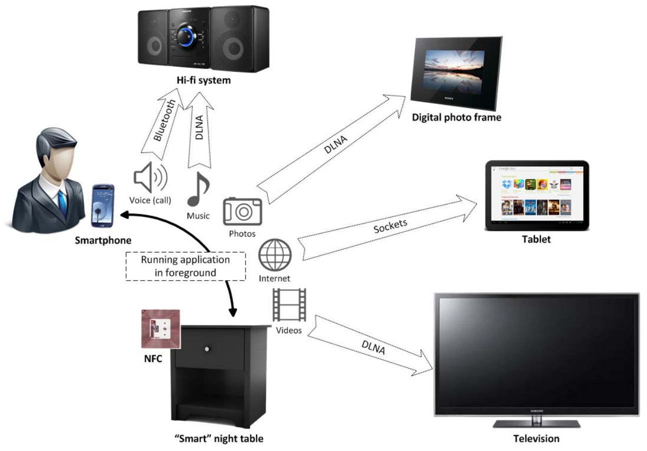


Figure 1. Overall view of the system.

The interaction method is described in Figure 2, through an activity diagram showing the workflow of stepwise activities, actions and choices. The key object of the case of use is the smart night table that triggers and coordinates the response of the smart environment depending on the applications that are running in the user's smartphone. The smart night table has been augmented with an NFC tag, thus when the smartphone reads the tag the process begins. If a user is listening to a playlist of his favorite music through his smartphone and he wants to continue listening the playlist through the audio system installed in the room, he only needs to place the phone on the night table. This event triggers an action that involves transferring the audio files from the mobile to the audio system in the room.

In the same way, if the user is playing a video or watching some photos through his smartphone, he can decide to transfer those contents to other available interfaces in the room, as a TV in the case of the video file, or a photo frame in the case of the album. Placing the smartphone on the night table will make a service to trigger the event of streaming the multimedia content stored in the mobile to a target suitable interface.

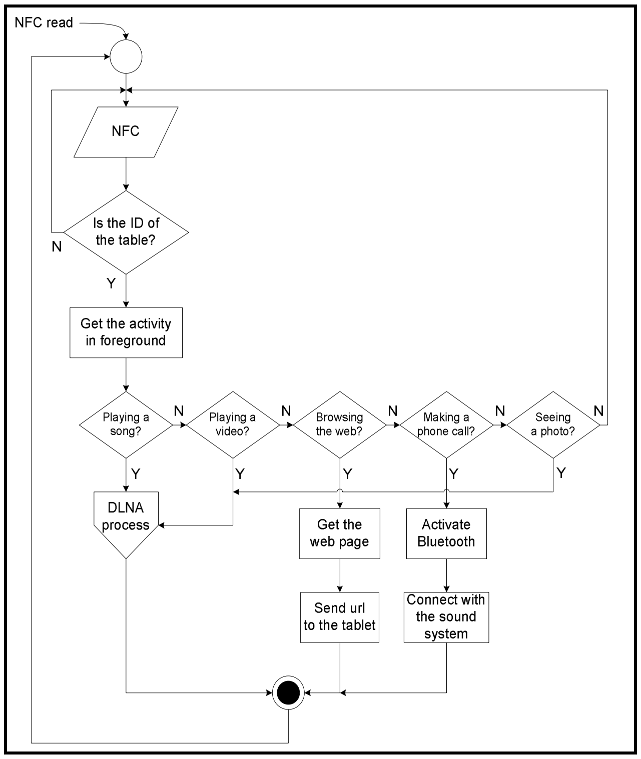


Figure 2. Instance of the interaction method for the Case of Use: a mediator night table.

Apart from the management of multimedia files, the user can be browsing over a web page that he wants to see more comfortably in a tablet device. Placing the smartphone with that web page opened in the mobile's web browser over the smart night table will trigger the desired action.

Additionally, if the user is talking to a friend and he wants to carry on the conversation while he gets dressed, he only needs to place again the phone with the active conversation on the night table, so the service in background detects the interaction with the night table, analyzes the current application running in the mobile and triggers the event that changes the current conversation from the mobile's speaker to the room's audio system.

TABLE I. LIST OF ACTIVITIES AND RELATED ACTIONS.

Activity in the mobile	Action
Playing a song	Playing the song using the *sound system*
Playing a video	Playing the video on the *television*
Browsing the web	Open the same web page in the browser on the *tablet*
Making a phone call	Continue the call using the *sound system*
Seeing a photo	Open the photo in the *framework*

Table I summarizes the list of actions related to the activities described above. The involved object for each action is marked in italics.

IV. MANAGING MEDIA CONTENT

Previously to a proposal of architecture to deal with the interaction concept described above, it is necessary to explore the technologies that enable media content management among devices, as this aspect is key for the case of use in Section III.b. Below we refer to our choices: the use of DLNA technologies and REST interfaces, which will serve as the basis for our architecture.

A. DLNA technology

The problem of transmitting multimedia contents between the smartphone and any other interface can be face by designing an ad hoc service or by adapting an existing technology. With this latter idea in mind, it is possible to consider different wireless streaming formats, such as Airplay Mirroring from Apple, Intel Wireless Display (most commonly known as WiDi), Wireless HD, Miracast (backed by the Wi-Fi Alliance) or DLNA. We have opted for DLNA [8], which stands out from other digital content streaming technologies thanks to its maturity, interoperability between different devices in different operating systems with different communication technologies, the support of main brands, thousands of certified devices and software on the market, and the availability of multi-platform SDKs to design new services and solutions.

DLNA stands for Digital Living Network Alliance. It defines a set of interoperability guidelines to enable sharing of digital media such as music, photos and videos between multi-branded consumer devices through IP based technologies such as Ethernet, Wi-Fi 802.11, Bluetooth, HPNA, MOCA and Wi-Fi Direct. The Universal Plug and Play (UPnP™) [9] Device Control Protocol is used by DLNA for media management, discovery and control. UPnP defines the type of device that DLNA supports and the mechanisms for accessing media over a network. The DLNA guidelines then apply a layer of restrictions over the types of media file format, encodings and resolutions that a device must support.

A main issue for our architecture is that DLNA defines different device classes. A device class specifies the functional capabilities of a device regardless of its physical attributes. A certified product must comply with all the requirements of its device class. These device classes are divided in two groups: home network devices and mobile handheld devices. In a typical DLNA scenario, the following actors can be found:

- Digital Media Servers (DMS) and Mobile Digital Media Servers (M-DMS) store content and make it available to networked and mobile digital media players, digital media renderers and digital media printers.
- Digital Media Players (DMP) and Mobile Digital Media Players (M-DMP) find content on networked and mobile digital media servers and provide playback capabilities.
- Digital Media Renderers (DMR) play content received from a networked or mobile digital media controller, which will find content from a networked or mobile digital media server.
- Digital Media Controllers (DMC) and Mobile Digital Media Controllers (M-DMC) find content on networked and mobile digital media servers and play it on digital media renderers.
- Digital Media Printers (DMPr) provide printing services to the DLNA network.

B. *REST interfaces*

Once the streaming of the multimedia content is solved, the problem focuses on the management of the different DLNA resources (servers, renderers, multimedia content stored in the servers) through an external service. Thinking in a more complex environment, any background service hosted in whatever system, regardless the operating system controlling such device, should be able to request, query, select and update the resources provided by the DLNA network. Such interaction can be solved in a light and efficient manner thanks to a RESTful API provided by the DLNA system.

The goal of the architectural principles that are core to the web, namely Representational State Transfer (REST) [10] is to increase interoperability for a looser coupling between the parts of distributed applications in a lightweight and simple manner seamlessly integrated to the web [11]. REST uses URIs for encapsulating and identifying services on the web.

In its web implementation it also uses HTTP as a true application protocol. This way, REST brings services “into the browser”: resources can be linked, bookmarked, cached, searched, and the results are directly visible within any web browser. Requests for services (i.e., verbs on resources) are formulated using a standard URI.

For instance, a GET HTTP method is used to request a resource identified by a unique URI. Similarly, a POST HTTP method is used to create a new resource, a PUT HTTP method to update a specific resource and a DELETE HTTP method to remove a resource.

Traditionally, REST has been used to integrate websites together. However, the lightweight and ubiquitous aspects of REST makes it an ideal candidate to build a “universal” API (Application Programming Interface) for embedded devices. This concept is often referred to as “Web of Things” [12], [13], [14].

Focusing on the case of use suggested, the background service running in the smartphone will make use of the RESTful API provided by the DLNA Digital Media Controller to request the state of the DLNA resources and to take actions over those resources, such as taking control of the renderers associated to the interfaces, playing multimedia files in those renderers and control the playback of those multimedia files.

V. Architecture Design

A. *Architecture scheme*

Regarding the DLNA devices for the case of use described, the proposed architecture is shown in Figure 3. Firstly, the smartphone should be equipped with a mobile digital media server where the user stores the multimedia contents that can be transmitted to other interfaces. It should also have a digital media renderer, where the media will be played, so the digital media controller can request the state of the renderer and the media content that is being played in the terminal.

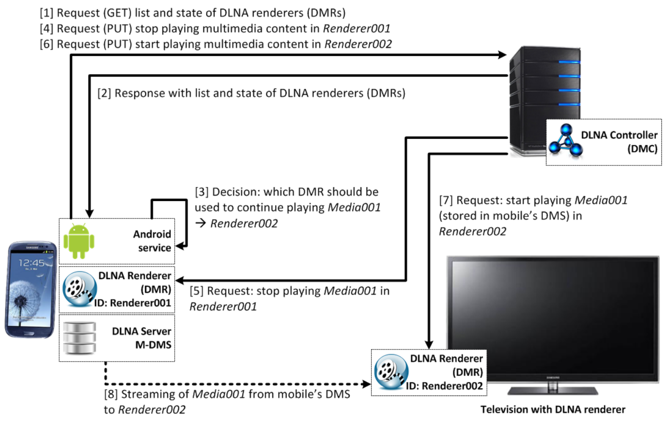

[1] Request (GET) list and state of DLNA renderers (DMRs)
[4] Request (PUT) stop playing multimedia content in Renderer001
[6] Request (PUT) start playing multimedia content in Renderer002
[2] Response with list and state of DLNA renderers (DMRs)
DLNA Controller (DMC)
[3] Decision: which DMR should be used to continue playing Media001 → Renderer002
Android service
[7] Request: start playing Media001 (stored in mobile's DMS) in Renderer002
DLNA Renderer (DMR) ID: Renderer001
[5] Request: stop playing Media001 in Renderer001
DLNA Server M-DMS
DLNA Renderer (DMR) ID: Renderer002
[8] Streaming of Media001 from mobile's DMS to Renderer002
Television with DLNA renderer

Figure 3. DLNA components.

Additionally, a digital media controller must be present in the network to have under control both the mobile digital media server with the shared contents and all the interfaces equipped with digital media renderers where the multimedia contents can be played.

Finally, all the devices where the multimedia content stored in the mobile digital media server could be transferred should be equipped with DLNA certified digital media renderers, so the digital media controller can generate the instructions to change the multimedia content from the smartphone to any other interface.

A background service running in the smartphone will be the responsible of communicating with the digital media controller to request the state of the network and to forward actions between the smartphone and the available interfaces through the same digital media controller.

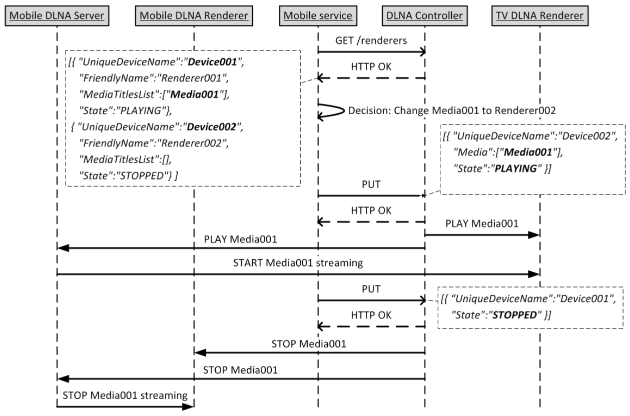
B. *Case-of-use workflow and implementation details*

The system in the case of use is thus composed by a smartphone, a table with a NFC tag, a photo framework, a sound system, a tablet and a television.

We use a television, a sound system and a photo framework with native DLNA support so we do not need to develop any component for these devices. The only component in which we need to handle the DLNA is a controller, designed to be REST compatible, which is stored in an external server.

Additionally, as the user may configure the system to transfer the websites being browsed in his smartphone to a tablet device, the tablet will need to have a service to receive the target URL to open. Thus we have developed a service that listens on a socket and launches an *intent* to open the default browser when it receives a URL from the smartphone.

The rest of components work in the mobile phone. They are the media player and the photo gallery with DLNA integration (render and server), and the service that handles all the process. The service also includes a component to detect incoming calls and switch the audio output between the phone speaker and the external sound system through Bluetooth.



```
[{ "UniqueDeviceName":"Device001",
   "FriendlyName":"Renderer001",
   "MediaTitlesList":["Media001"],
   "State":"PLAYING"},
 { "UniqueDeviceName":"Device002",
   "FriendlyName":"Renderer002",
   "MediaTitlesList":[],
   "State":"STOPPED"} ]
```

```
[{ "UniqueDeviceName":"Device002",
   "Media":["Media001"],
   "State":"PLAYING" }]
```

```
[{ "UniqueDeviceName":"Device001",
   "State":"STOPPED" }]
```

Figure 4. System workflow.

As it is explained in Figure 4, the interaction always starts with the reading of a NFC tag attached to a trigger smart object. This is the only event that is able to wake up our service. In our case of use, the tag will be associated with the night table, thus other readings will be discarded. Then the process can evolve in different ways depending on the activity (applications/services) enabled in the mobile device.

The simplest case is the incoming call. In this situation the service searches the Bluetooth address of the sound system and if it is available it connects to it. From that moment the audio is automatically passed to the sound system and the user can listen to his interlocutor directly through the sound system. This is the simplest case because it does not need any special component, as the operating system offers everything we need.

Another relatively simple case is browsing on Internet. In this case we suppose the user is browsing using our application. The service detects the web page open in the browser and copies the URL. Then the service connects to the tablet using a standard TCP/IP socket over IEEE802.11 wireless connection and it sends the URL to the service on the tablet. This communication uses a normal client/server model where the service on the smartphone is the client and the service on the tablet is the server. As soon as the tablet receives the URL it opens the default browser at that page. The use of anchors is also possible to open the web page at a more precise point.

The other cases are more complicated because of the lack of a native support of DLNA in Android. In these cases (audio, video and photos) we use an external controller to store and propagate the list of available renders and to control the streaming. This interaction uses a RESTful architecture and it is shown in Figure 3. Let us consider the transfer of a video playing in the mobile device. First, the mobile needs to know the devices it can use. They are identified by their DLNA renders. The service connects to the server (DLNA controller) and gets a list of renderers and their state (GET /renders). With these lists the service can know if some renderer is showing something and it can decide to stop it and change what it is showing. Then, the service chooses the renderer according to the activity open in foreground in the mobile. When the controller is ready it enables the renderer and tells the DLNA server in the smartphone that it can play the video on the chosen renderer. The service can change the state of the renderer, for example if the user wants to stop the video. To do that the service communicates a change of state to the DLNA controller using the PUT method. The DLNA controller stops the video on the DLNA server and it stops the streaming on all devices.

This workflow for DLNA devices is the same for video, audio and photos since the DLNA does not differentiate the content of the media. It is responsibility of the service to determine the renderer according to the content open in the smartphone.

We have implemented the mobile client in a Galaxy Nexus smartphone based on Android 4.2 with NFC reader. To detect the trigger event, we use a NFC tag placed on top of the night table: when the user puts the phone on the table, the tag is read. The value of the reading is captured by a service that checks the foreground application and starts the DLNA process. Since Android does not have a native support for DLNA we had to implement our application to manage the various DLNA devices.

VI. Conclusions and Further Work

This paper describes a mobile-instrumented interaction method that enables the transfer of media content and services in the mobile device, through an architecture that enables orchestration of the smart objects. A particular and interesting type of smart objects is that gathering devices with visualization and media capabilities. We are confident that a confortable level of personalization of the smart environment can be partially achieved if we are capable of providing a simple way of transferring personal content from mobile devices to other smart objects through physical interactions.

To validate our approach, we describe and detail the architecture for a specific case of use that we have implemented. It is based on a trigger object, a night table in a smart room, which initiates customized actions among the different elements in the space.

The concept is simple and powerful, and we think that it enables easy and seamlessly interaction with the smart environment. We are working in improving it by extending the set of actions or movements the user can make when interacting with the smart object that triggers the action, and also studying the remainder workflow when the user continues to use the smartphone, i.e. takes the phone from the smart table to have a look at something, but he does not want to interrupt the action that has been already configured.

Additionally, we are enhancing the interaction method to define how the user may interact with the multimedia smart objects once the content or the transfer controls have been dispatched. For example, we are aim at satisfactorily configure the system response if a user wants to turn up the volume of the video once it is being played in the TV, as we do not want him to take his phone again to do so. Instead, he can interact with the TV through some gestures with significant meaning, that will be interpreted by a device such as a Kinect, which will translate those gestures to actions to be taken (change the volume, skip to the next photo or music file…).

Another pending issue in our current implementation is the integration of context recognition to enable the system with more advanced capabilities to adapt its actions. In the described case of use, this means, for example, being capable of using the room occupancy information to adapt the media responses.

Acknowledgment

This work has being supported by the Government of Madrid under grant S2009/TIC-1485. The authors also acknowledge fruitful related discussions within the THOFU initiative, financed by the Spanish Center for the Development of Technology (CDTI).

References

[1] Kawsar, F., Fujinami, K., Nakajima, T. 2008. Prottoy Middleware Platform for Smart Object Systems. *International Journal of Smart Home*, vol. 2, no. 3.

[2] Siegemund, F., Floerkemeier, C., Vogt, H. 2004. The value of handhelds in smart environments. *Procs. of ARCS*, LNCS 2981, pp. 291-308.

[3] Rukzio, E., Broll, G., Leichtenstern, K., Schmidt, A. 2007. Mobile Interaction with the Real World: An Evaluation & Comparison of Physical Mobile Interaction Techniques. *Procs. of AmI 2007*, LNCS 4794, pp. 1-18.

[4] Want, R., Fishkin, K.P., Gujar, A., Harrison, B.L. 1999. Bridging Physical and Virtual Worlds with Electronic Tags. *Procs. of the Conf. Human Factors in Computing Systems*, ACM Press, pp. 370-377.

[5] Beigl, M. 1999. Point & Click – Interaction in Smart Environments. *Procs. of the First Int. Symposium on Handheld and Ubiquitous Computing*, LNCS 1707, Springer-Verlag, pp. 311-313.

[6] Broll, G., Paolucci, M., Wagner, M., Rukzio, E., Schmidt, A., Hubmann, H. 2009. Perci: Pervasive Service Interaction with the Internet of Things. *IEEE Internet Computing*, vol. 13, no. 6, pp. 74-81.

[7] Bernardos, A.M, Casar, J.R., Cano, J., Bergesio, L. 2011. Enhancing interaction with smart objects through mobile devices. Proc. of the 9th ACM International Symposium on Mobility Management and Wireless Access, Miami (USA), 31 October - 4 November 2011.

[8] http://www.dlna.org

[9] http://www.upnp.org/

[10] Fielding, R.T., Taylor, R.N. 2002. Principled Design of the Modern Web Architecture, ACM Trans. Internet Technology, vol. 2, no. 2, pp. 115-150.

[11] Guinard, D. et al., 2010. Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services", IEEE Transactions On Service Computing, vol. 3, no. 3, pp. 223-235.

[12] Wilde, E. 2007. "Putting Things to REST," Technical Report UCB iSchool Report 2007-015, School of Information, Univ. of California, Berkeley.

[13] Guinard, D., Trifa, V., Pham, T., Liechti, O., 2009. Towards Physical Mashups in the Web of Things," Proc. IEEE Sixth Int'l Conf. Networked Sensing Systems (INSS '09), pp. 196-199.

[14] Luckenbach, T., Gober, P., Arbanowski, S., Kotsopoulos, A., Kim, K. 2005. TinyREST—A Protocol for Integrating Sensor Networks into the Internet," Proc. Workshop REALWSN.