Virtual Objects on the Internet of Things

Jordán Pascual Espada, Oscar Sanjuán Martínez, B. Cristina Pelayo García-Bustelo,
Juan Manuel Cueva Lovelle.
Computer Science Department, Oviedo University, Oviedo, Spain

***Abstract* — As technology advances more and more "things" began to appear in digital format, such as: tickets, agendas, books, electronic purses, etc. Internet of things encourages communication and integration of physical objects with each other and people to automate tasks and improve efficiency. Digital objects like physicists should be part of Internet of Things but the different structures of these digital objects causes in most cases these digital objects can interact only with specific applications that know the specific format. Based on the problems in this paper proposes a structure that supports the generic construction of virtual objects irrespective of their business logic and their integration with other applications and "things".**

***Keywords* — Virtual Object, Internet of Things, DDTS, Web Services, Smart Phone.**

I. Introduction

THE main idea in the Internet of Things is that any "thing" or object, conveniently tagged, may be able to communicate with other objects equally tagged through internet or any other protocols. These objects which are part of the net may contain small chips or embedded systems, depending on their purpose [1]. They may range from home equipment to industrial items or even electrodomestic, cars or even supermarket food. Anything can be tagged to be part of the Internet of things [2, 3].

Possibilities of the Internet of things to make people's life easier and to automatize many of our current tasks are huge, for instance, it is possible that the fridge may send an email to our mobile phone if it runs out of milk, we can monitor hospitalized patients by internet... there are lots of practical applications and all of them are seen with a common basis: "things" are communicating with "things" or persons. [4]

Parallel to the development of technology, more and more objects called "things" which are merely physical start to be seen also in digital format. Examples of them can be seen in: books, maps, e-tickets for gigs, plane tickets, agendas, contact cards, agendas, electronic purses etc.

Not all "virtual objects" that are used today are digital models of physical objects, sometimes these objects are new concepts designed for any type of task or information encapsulation. We have defined the term "virtual object" as a digital element has a specific purpose, comprised a series of data and can perform actions.

This work was supported in part by the Oviedo University under Grant IOVD-01. A.

When we observe the behavior of these virtual objects we see there is no standard format or any recommendation to normalize their usage. There is no mechanism by which we can treat them in a general way, store them, share them or process them with other applications which may not know their format.

Problems coming out of this lack of standard format are the following:

- Difficulties for decode: devices with no specific applications to decode the virtual object will not be able to process it. Let's take as an example my Mobile phone; if I transfer a contact card to another user, the Mobile intended to receive it won't be able to decode the incoming information. This handicap leads to the need of installing many applications in case we want to operate with different virtual objects. It makes it harder for a company or developer to place in the market their own virtual object, since nobody would be able to decode it without the suitable software.
- Lack of Communications: Ideally, the objects linked to the Internet of things to interact among themselves and with other applications to automate tasks and increase efficiency [5]. Since there is no standard format way to get actions or services from a giving virtual object, it is very difficult to interact with another application. Let`s illustrate it with a cinema ticket which is basically related to being a mere number code with stored information in a company database. The ticket is decoded by a web application and a specific machine. By focusing on this, it is very complicated for a virtual object to directly communicate with other applications or to transfer the ticket to other user.

Internet of things follows the aim of making the Communication between things possible, so things can communicate by themselves with other things and users. A physical thing may have a catalog of actions which is used to communicate, for instance a sensor connected to the net offering service to get position and temperature. The focus of something connected to a web which has a catalog of actions and is able to communicate by itself with other users is crashing frontally against the focusing of virtual objects, which do not exist themselves, independently, as entities, but only to form part of an application which interprets them.

This work has been divided into the following sections: In

DOI: 10.9781/ijimai.2011.144

the second section provides an analysis of current trends in systems based on virtual objects. In the third section we define the objectives to be met by the proposed model. In the next section we examine virtual objects in order to define their features and get their points in common. In the fifth section presents a proposal for a common structure to support the construction of virtual objects. In the following two sections we construct a prototype virtual object with the proposed structure and show the operation of the prototype. At the end of the document refers to: potential uses of virtual objects and standardized research findings.

II. Current Models of Virtual Objects

At present there are few systems that deal elements that could be considered as virtual objects, mainly there are two different approaches to model them.

A. Resources managed by device applications.

In this type of system virtual objects are composed of a number of records stored in a database. The records are dependent on an application that manages the application contains the business logic that is run against the records in the data store. The information associated with every record that represents a virtual object varies according to each system and business logic (Figure 1). Sometimes there may be rules or conventions to model virtual objects belonging to specific business logic, but there is no general model.

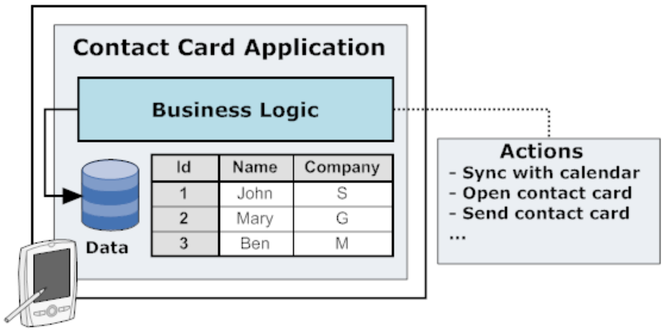


Figure 1. Structure of a management application contact cards.

Localized deficiencies in these systems are:

- One of the objectives is that virtual objects run in a variety of devices. Devices that want to play a virtual object need to have installed a specific application that recognizes the particular format of the object.
- Derived from the first problem devices requires a large number of installed applications, if each application is only able to interpret a particular type of object. In addition to the complication posed by developing an application for each type of virtual object, device and operating system. It would be impractical for every website need a specific browser to be interpreted.

Following the approach of internet of things, the virtual objects should also be able to integrate with other applications, devices and users. When a virtual object is designed as a resource which is treated by a specific application difficulties arise that can be accessed or discovered in a generic way by another application or device. Depending on whether the application managing the virtual object mechanisms offers some kind of integration or other applications may access the object. In an ideal situation a specific virtual object should be able to be accessed by other objects or applications, and should be able to do the same. Communication with the virtual objects should be conducted in a standardized way, although obviously each object will not make the same actions but how to display and access to those shares if it should be common to them all.

B. Resources managed by Web applications.

In such systems, virtual objects are records in a data warehouse and managed by a web application. Users interact with virtual objects using web browser, in some systems also use specific devices such as ATMs (Automated Teller Machine, cash machine) that are connected directly to the management application (Figure 2). This approach gets to the interpretation and management of the object is not conditioned by the applications installed on the client, as it is done through a web browser.

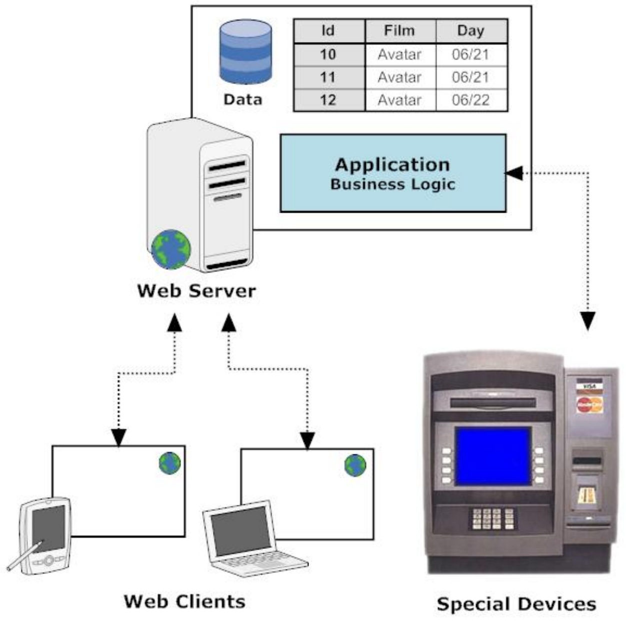


Figure 2. Structure of an application for the sale of cinema tickets.

This type of system continues to make difficult the possibility that virtual objects can communicate with other applications or users outside the application where they are running. Currently there are several web applications which provide APIs as web services for data that are stored in the web application (that model an object) can be integrated applications. Although this alternative may be sufficient in some cases, this solution is far from ideal. There are still difficulties for the virtual object could communicate with a physical object nearby, who holds the location-dependent services [6] (supermarket, parking, etc.). Although the Web application has access APIs, is difficult for a virtual object hosted in a web application takes the initiative in interacting with other Things or application.

III. Objectives

In this document we will give shape to a possible format recommended for the construction of virtual objects. Main objectives are as follows:

- The proposition of a common structure for the construction of virtual object, in which all of them, regardless complexity or business logic, can be: interpreted equally by any electronic device which is provided with the computational capacity needed (enclosed systems, computers PDAs, mobile phones etc) without the need of any previous configuration or specific software.
- To favour the integration and communications of any virtual object with applications and users. It will be similar to the process followed to integrate physical elements to the internet of things since virtual objects should offer the choice of discovering them to other users or applications, as well as getting their action and service's catalogs or even interacting with them.

Another objective is that the designed solution is a technologically possible with the systems and devices that exist today and that also is consistent with the trends and evolution that follows Internet of things.

IV. Features of Virtual Objects

A. Interaction levels

The level of integration that can have a virtual object on the internet of things is difficult to determine because it can cover a lot of cases. Physical objects have different levels of integration, for example, a yogurt has an RFID tag [7] that is read by other things such as a smart refrigerator. The intelligent fridge is an object much more "complex" than the yogurt, this object has a catalog of actions that enables it to interact with people or other things such as RFID tagged food.

As physical objects, virtual objects can have different degrees of integration depending on its purpose. For virtual objects could have a similar integration of a complex physical object connected to the Internet of things, the structure of virtual objects should provide the opportunity to discover objects and their associated actions.

The structure of virtual objects should include several levels of interaction, and also be able to model virtual objects with a lower degree of interaction, for example, virtual objects that only have to be read or processed by other applications or devices (similar to happened with the yogurt).

In conclusion, although a specific virtual object may not require all types of interaction, the model should support the following types of interaction:

- Interaction with people. A person must be able to interact with the virtual object and control it through an electronic device. Depending on the purpose of the virtual object could be users with different privileges to control some actions in the object (Figure 3). In the same way that a physical object, a virtual object should be able to change owner, this property could involve a device migration.

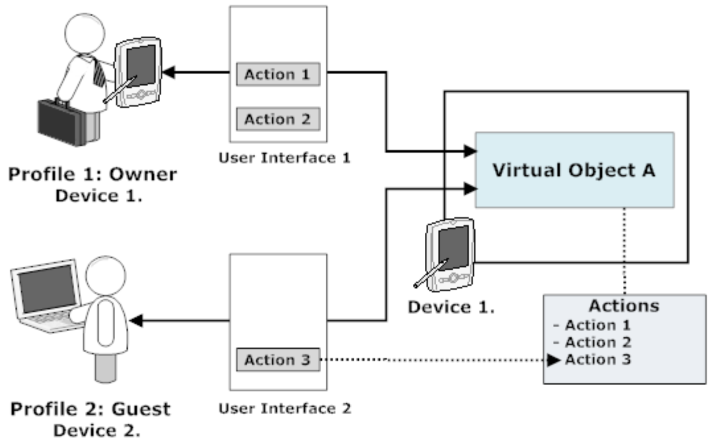


Figure 3. Structure of an application for the sale of cinema tickets.

- Interaction with applications / things. This type of interaction can be divided into two parts (Figure 4):
  1. Interaction as a transmitter: the virtual object has to be able to initiate interaction with another thing or application.
  2. Interaction as a receiver. The virtual object must be able to be discovered by other "things" or applications. Once discovered the virtual object, things can interact with the object invoking actions that have been in their catalog.

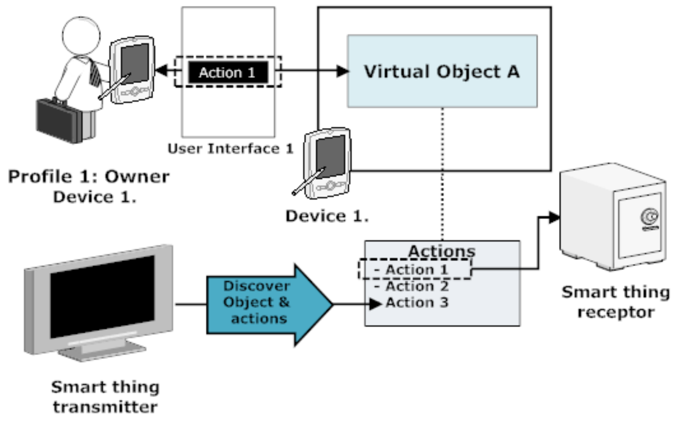


Figure 4. Structure of an application for the sale of cinema tickets.

B. Structure and properties

It deals with the searching of a unique structure which may lead to the rebuilding, in the same way, very different objects: plane tickets, intelligent publicity, contact cards etc.

In a similar way of a conventional application, parts of a given virtual object could be divided in three layers:

- Application layer, in which we include the needed mechanisms so the virtual object can interact with users and applications. The classic form of interactions with users is by means of a rich graphic interface. The interaction with other applications is normally made through a service catalog.

- Business logic, in which we found all the Business logic, executable coded or services the object can carry out.
- Data access layer, in which necessary data are stored in order to operate with the virtual object.

The design of the structure could be similar to that of a conventional application, but still underlies a great difference in their natures. Frequently, virtual objects are downloaded through internet or are transferred from computers and this is the reason why they should offer a structure very capable of migrating between devices in a very dynamic way, very lightly, and with no installation required. Instead of settling should be run in a “sandbox” with limited permissions [8].

Generally, all objects should have a series of common properties which may allow identification: name, type and identification.

At the time of dealing with an object, there are common actions which are similar to those of a file. Can they be copied? Can they be modified? Can they change owner, or be transferred?

All needs and observations commented, have been taking into count when it comes to elaborate a common structure for virtual objects.

V. PROPOSAL: STRUCTURE OF VIRTUAL OBJECTS

Being based upon detected needs, this proposal defines that the structure of a virtual object is formed by a group of files of different nature. The precise model proposed in this document has been called Virtual Objects DDTS (Device Dependent Temporary Services). The choice of this name is based on the following arguments:

- Services. The virtual object will communicate with other devices or applications sharing the actions it associates for this purpose use a catalog of services.
- Temporary. Most of the virtual objects are not designed to have a persistent character as an ordinary application. The proposed structure attempts to model objects "supplies" which are used in a limited period, until a certain date or activity.
- Device Dependent. The virtual object needs to be placed in an electronic device, which acts as a container of objects. One of the properties of these objects is to be able to change host device to maintain its status and function. Sometimes the objects will be downloaded from the internet; others migrate between devices, etc.

The elements that form the structure virtual object DDST are:

- Descriptor: XML file, which contains information about identity, configuration, general behaviours, arrangement and execution of the virtual object.
- Graphical interfaces: XML files, each one represents a screen, which is the means by which the user can communicate with the logic of a virtual object.
- Service catalogs: there are resources which have the function of showing the applications or programmes, the actions an object may carry out. This interaction is achieved by means of service catalogs, which execute actions in the business logic of the object.
- Executable code: a file which contains the needed code to execute the virtual object logic. The code can be obtained in different formats or languages of programming in order to be executed in devices of different characteristics.
- Data storage: if the logic regarding Business services may require persistence, this must be provided with a file responsible for the arrangement of that information. Virtual objects are relatively simple models when compared to that of a conventional application. They are not thought to store a great deal of registers but only a few values.
- Additional resources: Could be included a Lumber of extra resources, in a non-limited way. Generally, this will be multimedia resources: images, icons and videos, which will be used to complete the graphical parcel regarding virtual object.

VI. DEVELOPMENT OF VIRTUAL OBJECTS DDTS

To illustrate the use of virtual objects DDTS has implemented a movie ticket, following the proposed specification; this prototype will be used to illustrate the structure and operation of virtual objects.

Complementing the virtual object input has developed an application "manager of virtual objects," which runs on the Android mobile operating system [9]. The ideal container for virtual objects is an electronic device that we carry with us all day, which allows us to interact with objects at anywhere. We selected the Android mobile platform for developing the prototype because it is open source and devices that have this operating system have characteristics of computing and communication technologies, these features are sufficient to ensure interaction with virtual objects.

This application is able to interpret and work with any virtual object built to specification.

A. *Executable Code - Business logic.*

In this model the actions associated with cinema ticket will be:

- Show Movie Info: title, synopsis, images, etc.
- Show information about ticket: film, cinema, etc.
- Validate ticket: validation of the cinema ticket at the intelligent door cinema.

The business logic is implemented in a code file as if it were a conventional application. The virtual object can contain multiple source files that implement the same business logic, so that the devices running the object select the appropriate code to run on the operating system. This prototype uses the

Java language for implementation, because we know that only runs on Android system. The implementation is done in standard Java by inheriting the VirtualObject specific class. The code will be dynamically loaded by the manager of virtual objects, so that methods to be invoked must be declared as public.

B. Data storage - Business logic.

Sometimes the business logic may require that some data have a persistent nature. Virtual objects are simple models, so they will not store many registers and will not require a traditional database. To support persistent data they are declared as key-value pairs in a specific file. From the executable code one can easy access to the stored values, using special methods (Code 1), which are implemented in the class VirtualObject. This system achieves a simple and efficient synchronization with the data store that is almost transparent to the programmer.

In the case of the cinema ticket that we are implementing the value "used" would be a persistent value.

```
// Validate Ticket.
public boolean useIt() throws Exception{
        // Data warehouse access . Key - used.
        boolean isUsed = loadDataBoolean("used");
        if(!isUsed){
            ...

 // Modify data warehouse . Key - used, Value - true
            saveDataBoolean("used",true);
            ...
}
```

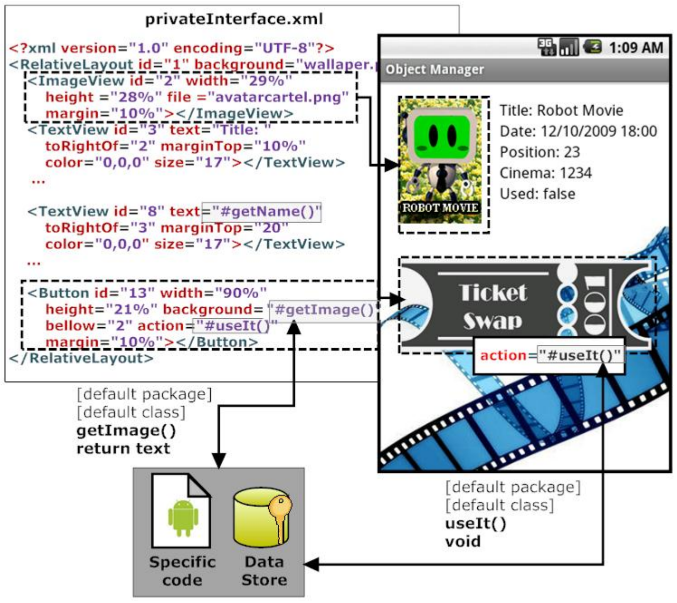
Code 1. Java code. Access to the data warehouse using specific methods contained in class VirtualObject.

C. Graphical interfaces - Interaction with people.

Graphic interfaces are the main form of interaction with the virtual object, its use is to provide a simple visual environment to enable communication between user and virtual object. The user interfaces should belong to one of two types:

- Private: used by the person owning the host device.
- Public: can be added if you want other users to discover and connect to the virtual object remotely.

Each XML file refers exclusively to a screen that can be displayed during performance of the virtual object. To describe the elements that appear on the screens and how they behave, we have started from a smaller version of the syntax used by the Android system [10], describing user interfaces in a relative way, so they can be interpreted in the same way regardless of the resolution or screen size of the device. The elements that compose the graphic interface, define its appearance and behavior using XML properties. These properties can refer to methods in the executable code. (Figure 5).



```
privateInterface.xml
<?xml version="1.0" encoding="UTF-8"?>
<RelativeLayout id="1" background="wallaper.
  <ImageView id="2" width="29%"
    height ="28%" file ="avatarcartel.png"
    margin="10%"></ImageView>
  <TextView id="3" text="Title: "
    toRightOf="2" marginTop="10%"
    color="0,0,0" size="17"></TextView>
  ...

  <TextView id="8" text="#getName()"
    toRightOf="3" marginTop="20"
    color="0,0,0" size="17"></TextView>
  ...
  <Button id="13" width="90%"
    height="21%" background="#getImage()"
    bellow="2" action="#useIt()"
    margin="10%"></Button>
</RelativeLayout>
```

```
[default package]
[default class]
getImage()
return text
```

```
[default package]
[default class]
useIt()
void
```

Title: Robot Movie
Date: 12/10/2009 18:00
Position: 23
Cinema: 1234
Used: false

Figure 5. Private graphical interface of the virtual subject: movie ticket.

The graphical interface displayed may be accessed and modified in the executable code using special methods implemented in class VirtualObject. With this functionality, developers can implement changes in the graphic interface as a result of actions (eg pressing a button).

In the case of the cinema ticket it has included a private graphical interface, through which the owner of the ticket could manage, and a public graphical interface that allows other users to connect to the virtual object entry and see an overview of film.

D. Service catalogs - Interaction with things/applications.

The virtual object can be accessed by other devices, applications or virtual objects. The object publishes its catalog of services in Web Services Description Language [11] and in a specific API for the integration of virtual objects.

E. Descriptor – Configuration.

The descriptor file is an XML document which contains information about the identity, configuration and implementation of the virtual object. The information contained is as follows:

- Identity Object: Name, Type, Unique Identifier (if it is unique) and icon.
- Behavior of the object: If it is transferable, copyable or editable, if you have an expiry date, etc.
- Interfaces: Name the main interfaces of the object.
- Executable code: Name of the source files that may exist, at least there must be one. May also include the name of the main class.
- Data: Name of file data store.

In the particular case of this movie ticket could be a valid configuration file (Code 2).

```
<?xml version="1.0" encoding="UTF-8"?>
<virtualObject>
    <name>Ticket: Robot Movie</name>
    <type>www.TicketVirtualObject.com</type>
    <id>47236271</id>
    <transferable>YES</transferable>
    <expiration>12-10-2009</expiration>
    <editable>NO</editable>
    <copy>NO</copy>
    <interface
        private="privateInterface.xml"
        public="publicInterface.xml"
        wsdl="interface.wsdl"></interface>
    <code
        android="robotMovie9780.apk"
        j2me="robotMovie9780.jar"></code>
    <data>info.obj</data>
    <mainClass>com.rmovie.Ticket</mainClass>
    <icon>icon_tiket.png</icon>
</virtualObject>
```

Code 2. Configuration file of the virtual object DDTS cinema ticket.

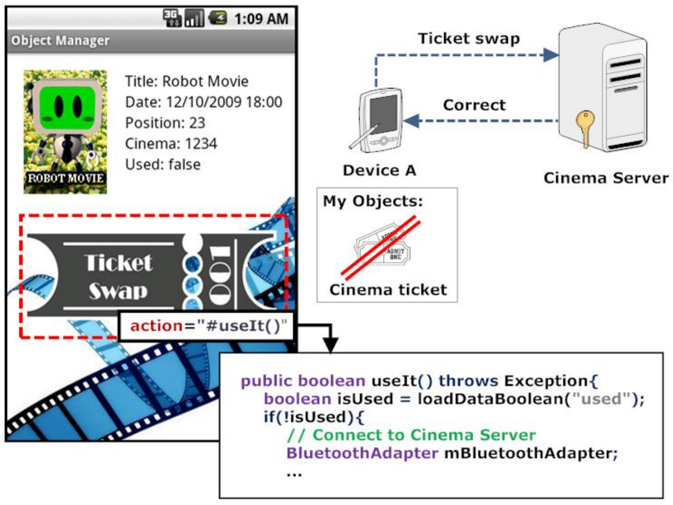
VII. Using Virtual Objects

Virtual objects DDTS are interpreted by an application which we call DDTS Manager, which is responsible for managing the virtual objects within a device. It must be designed to be installed and run on the given device, taking into account their characteristics, operating system, or programming languages it supports. Objectives of the manager are:

- Load, interpret and run any virtual object that has been built following a proposed structure. The first step to start using an object is to pick a manager, selecting the object configuration file. Once loaded the user can start interacting with the object, the manager interpret their interfaces and execute the corresponding code.
- Store and manage virtual objects. We will often use multiple virtual objects simultaneously; the manager must provide mechanisms to store and manage. Virtual objects that were loaded on the device must be displayed in an orderly manner to the user, so that it can interact with them and manage them, that is: delete, copy or transfer provided that the nature of the subject permits.
- Publish virtual objects and service catalogs. It can be specified in the logic of the object that allows execution of remote or publish their services, the manager has mechanisms to support such protocols relying on Bluetooth (Bluetooth, 2009), or other protocols such as: Internet, wireless etc.
- Discovery and remote execution. The manager will be able to discover the virtual objects that other devices publish. Once discovered the virtual objects may be performed remotely

A. Local execution

After the file of the virtual object is loaded into the manager, you can open it. The outcome of the interpretation of the virtual object is the main graphical interface through which the user launches events involving invocations of the shares included in the executable code of the virtual object. The most important action of the object movie ticket is validated against the server of cinema (Figure 6).



```
public boolean useIt() throws Exception{
    boolean isUsed = loadDataBoolean("used");
    if(!isUsed){
        // Connect to Cinema Server
        BluetoothAdapter mBluetoothAdapter;
        ...
```

Figure 6. Local execution. The event “touch button” causes the execution of the action: validate the movie ticket.

B. Remote execution

Depending on the logic of the virtual object DDTS is possible that this has to be used remotely by other people besides the owner. The device operates as a virtual object server, allowing other users to connect to it and run the virtual object remotely.

To initiate the remote execution on virtual objects must be "active" on the server device and be discovered by other client devices using Bluetooth, Wi-Fi or other protocols. The client device receives and interprets the main public graphical interface. Each time the client starts an event; the server receives a message and run the Corresponding code. The execution on the server can lead to changes in current public graphical interface as a result the server sends the new interface for the client device. The operation is similar to a web application server.

In this case the virtual input object accepts the possibility of being discovered and accessed by other users. Remote access is started discovering the virtual object on the device server, and then the client interprets the main public graphic interface, which contains images and the synopsis of the movie (Figure 7).

C. Virtual object behavior

The movie ticket has been modelled with the properties: unique and transferable, therefore can be atomically transferred between devices. The owner of the virtual object can use DDST manager to find a device which transfers the virtual object. The movie ticket will continue to have the same functionality and state after device migration.

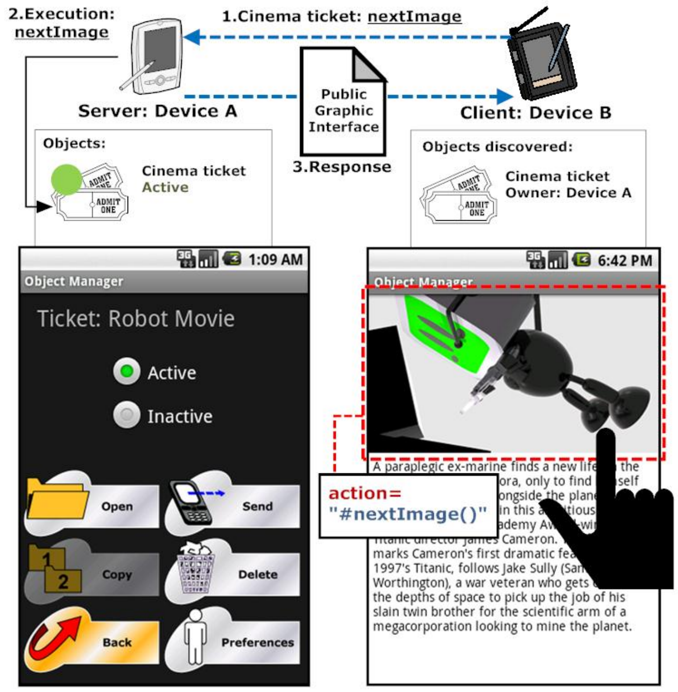


Figure 7. Remote Execution Process.

VIII. Potential Uses

Nowadays, many objects, either real or virtual, are good candidates to be re-designed as virtual objects. Thanks to these changes we could an improvement in the lifestyle of people, making it easier and automatizing many daily life tasks. Some examples are the following:

- Tickets: cinema, train, parking, etc. The proposal offers the possibility of storing them, for instance in our Mobile or PDA, as if it was our digital Purse.
- Multimedia objects: product catalogs, intelligent publicity, contact cards. The structure proposed offers the possibility of creating Rich interfaces, including lots of control, in sound, videos etc.
- Application generated resources: Shopping lists, events and schedules, since it contains business logic and services, a list cannot be limited to a series of numbered products, it can be also provided with certain intelligence and capacity to interact with other applications or elements.
- Remote control: remote execution gives the ability to communicate with devices that have been programmed public interfaces, whether or not virtual objects. Thus a single device can control a large number of elements.

IX. Conclusion

The benefits of the proposal on the current solution in the modeling of virtual objects are:

- It unifies the way we build virtual objects with a concrete structure, Strong enough to model complex virtual objects.
- It is designed so devices with the suitable computation ability can be executed or correctly interpreted in any normalized virtual objet, it doesn't matter the object logic or the device characteristics, (operative system, resolution...).
- Easy development, using languages of general purpose and widely extended formats for the construction of virtual objects. It offers automatic support to main properties which may define the object behavior.
- Strengthens and makes it easier the Communications between virtual objects, users and application, the same with the transfer or interchange of virtual objects.
- Virtual objects may have more ability to interact using communication mechanisms and specific recognition of mobile phones as cameras [12], GPS, sensors, etc.

References

[1] Kranz, M. , Holleis, P., & Schmidt, A. (2010). *Embedded interaction: Interacting with the internet of things*. IEEE Internet Computing, v 14, n 2, (pp 46-53).

[2] Kortuem, G., Kawsar, F., Sundramoorthy, V., & Fitton, D. (2010). Smart objects as building blocks for the internet of things. IEEE Internet Computing, v 14, n 1, (pp 44-51).

[3] Lu, Y.(Ed.), Yan, Z .(Ed.), Laurence,T. Y.(Ed.), Huansheng ,N. (Ed.). (2008). The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems. (Ed.), Auerbach Publications, Taylor & Francis Group.

[4] Global Trends 2025: A transformed world. (2008). Appendix F: The Internet of Things (Background). SRI Consulting Business Intelligence.

[5] Spiess, P., Karnouskos, S., Guinard, D., Savio, D., Baecker, O., Souza, L. M., & Trifa, V. (2009). Soa-based integration of the internet of things in enterprise services. IEEE International Conference on Web Services, ICWS 2009, (pp 968-975).

[6] Fajardo, R., Miramá, V., Caicedo, 0. Mesa, J.& Martínez, F.O. (2008) Descubrimiento e interacción de servicios móviles ubicuos utilizando Bluetooth y Wi-Fi.

[7] Roger, S. (2005). RFID: A Brief Technology Analysis, CTO Network Library. http://www.rfidconsultation.eu/docs/ficheiros/RFID_analysis.pdf.

[8] Yeon-Seok,.K., & Kyong-Ho,L. (2007). A Light-weight Framework for Hosting Web Services on Mobile Devices. Proceedings of the 5th IEEE European Conference on Web Services, ECOWS 07, (pp 255-263).

[9] Android . (2010). From http://developer.android.com/

[10] User Interface Android. (2010).From http://developer.android.com/guide/topics/ui/index.html.

[11] WSDL w3c Web Services Description Language. http://www.w3.org/TR/wsdl.

[12] Rohs, M., Gfeller, B. (2004).Using camera-equipped mobile phones for interacting with real-world objects. Advances in Pervasive Computing, Austrian Computer Society (OCG).