Using Model-driven architecture principles to generate applications based on interconnecting smart objects and sensors

González García, C., & Espada, J. P. (2014). Using Model-Driven Architecture Principles to Generate Applications based on Interconnecting Smart Objects and Sensors. In V. G. Díaz, J. M. C. Lovelle, B. C. P. García-Bustelo, & O. S. Martinez (Eds.), Advances and Applications in Model-Driven Engineering (pp. 73–87). IGI Global. https://doi.org/10.4018/978-1-4666-4494-6.ch004

Using Model-driven architecture principles to generate applications based on interconnecting smart objects and sensors

Cristian González García
University of Oviedo, Oviedo, Spain
Jordán Pascual Espada
University of Oviedo, Oviedo, Spain

ABSTRACT

Actually there has been a rise in the quantity of Smart Things present in our daily life: Smartphones, smart TVs, sensor networks, smart appliances and many other home and industry automation devices. The disadvantage is that each one is similar but very different of others because they use different kinds of connections, different protocols, different hardware and operative systems; even if they belong to a similar kind, and depending on the manufacturer, two Smartphones or two humidity sensors can be totally different. In spite of this, they all have something in common: All can be connected to Internet. This fact is what gives power to objects, because it allows them to interact with each other and with the environment by intercommunication, a joint and synchronized work. That's why the key resides in creating connections among different objects. This requires technical and qualified personal, although the majority of people who use it do not meet these conditions. This people requires objects that are easy to interact with and with as many automations as possible, for example, so they can use all the house's devices, like a cellular, or well that messages of situation (temperature, humidity) of home arrive to it, and to can order from a distance or let it automatized.

INTRODUCTION

Actually, most of the people have Smartphones or other intelligent mechanisms, like Smart TVs, Smart Labels (NFC, RFID)... This opens new doors to know the Internet of things.
Thank to these mechanisms, users can receive totally personalized notifications almost instantly, through an application, a message, a call, a code or by consulting a web service. Also, this gives the user the possibility to interact with other mechanisms in a much more easy way. For example, he can send a photo from his tablet to a Smart TV with a simple movement of the photo, 'throwing' it towards the Smart TV. All this give both the user and the developer endless possibilities. In our case, with a view to process automatization, notifications, references and configurations. All this is achieved by the interconnection and identification of objects using their different sensors and protocols. This way, objects can interact among themselves, with or without the user's interaction, but always performing tasks in an easier way or even automatically, according to the user's preferences.
For example, if this is applied to food, information about it can be given to other mechanisms in a simple way. A fridge would be able to send mails to the cellular phone, informing about the amount of food stored inside or the products that are about to expire, thanks to the reading of the information (Rothensee,

2007). All this is possible if the food has a Smart Label (NFC or RFID) and the fridge has a reader, a computation unit that can perform the suitable actions and an Internet connection (Gu & Wang, 2009).
In the same way that it is applied to a fridge, this technology can be used with other elements, as the house itself and its conditions when we are outside, cities, supermarkets and shops to improve the way to bring their inventory or give information to their clients… all that is needed is to joint those smart object by using the suitable sensors.
Internet of Things doesn't just present small-scale utilities at houses and shops but there are also some systems and IOT initiatives that include buildings and even integral cities.
Madrid, Santander, Málaga, Barcelona, Luxembourg, Aarhus, Turku, Beijing... (Vienna, 2013) All of these are smart cities (Hao, Lei, & Yan, 2012) and they use sensors and others smart things in order to perform different tasks.
Humidity, temperature, ozone, movement, pressure capacity, gas, noise, light, pollution… There are different kinds of sensors that can obtain real-time information about the city's condition.
Traffic, parking, timetables about Smart city transport (Falvo, Lamedica, & Ruvio, 2012), environmental danger, lack of trash recycling, the quality of the water, light control, traffic control, reduction of CO_2, energy saving, access to hospitals…All this is possible thanks to the combination of Smart Object and sensors in the cities.
For example, when a certain temperature is reached, the air-conditioning can be activated or the windows can be opened if it is not windy or rainy outside. Also, this information can be sent to Internet and the users within the affected zone can be notified by sending information or using smart labels that send the information to users with nearby compatible mechanisms.
By using other sensors (like those that react to movement) and the consignment of information to a process station and the cloud, both the traffic and the parkings can be controlled, and they also can offer or send information to the citizen, who will decide how they want to move.
In order to create a smart city, not everything has to be based on ordinary things, private things, can be also included. Also, each citizen can provide several sensors and install them in different places of their houses. For example, temperature and humidity sensors on a balcony. Other citizen could as well get access to this data if it is public and, following the same system, a page where the city map is selected, it is possible to observe the exact temperature and humidity in each zone. All this could extend and, with the use of the fridge mentioned before, know when certain element is needed (like water in a drought) and so the country could establish measures for any kind of situation.
This brings one problem: not all people has knowledges about sensor nor computers to be able to create the program that is needed for interconnection. Because of this, it's necessary to offer a system that can make this difficult task accessible for persons without specific knowledge about sensors and computing. It is necessary to offer it to persons with knowledge about language command, who know what they want to do and, thanks to a simple assembly and a little, easy configuration, can offer the information that they wish.
However, applications created for this case or other similar to this one, as is normal, share some identical program points, they treat similar data and they behave in similar ways. But in spite of that, they are applications created by different people so in the end the inside is different too.
This will result in the existence of several applications that are practically the same and aim to resolve the same problem, but they are used in a different way. Also, they could have been developed in a different way and each one has required a great amount of time for that development.
However, the most important thing for users is to have a great speed and automatisms that make easier to perform a certain task. In this case, connecting the sensor and registering it.
This way, it arose the Model-driven engineering. This fact offered the necessary abstraction layer to create some languages of specific control and improve the usability and accessibility for the creation of applications for those people unconnected to the computer world or just making the use of certain technologies easier. They only need to know the domain.

The productivity and speed were increased in those areas in which the Model-driven engineering was applicated.
In cases like this, the creation of a sensor net with its configuration and the connections with other mechanisms or smart objects is an arduous task. The diversity of objects (telephones, computers, smart labels, sensors…) and the different kinds of connection (Wi-Fi, Wireless, Bluetooth, SMS,…) make administration very difficult. All this, along with the different kinds of sensors (humidity, temperature, pressure, speed, movement…) makes this application, very hard to use for most of the people.
Because of this, the objective is the application of the Model-driven engineering to the field of sensors, and, by doing so, being able to form and create sensor nets easily and allowing the user to choose what objects he wishes to interact with and the actions that he wanted to perform in a clear and easy way.
In this way, we get to reduce the complexity thanks to the layer abstraction that is inserted to represent a specific domain. As well, the portability, interoperability and reusability created by it are improved.
The objective of this research is the presentation of the basics and the idea that will serve as a base to create the next prototype. This one will consist of a platform web in which smart object and sensors can be registered through the use of Domain Specific Language (DSL) done with Graphical Modeling Framework (Kolovos, Rose, Paige, & Polack, 2009) and Textual Modeling Framework.
With the creation of this platform, Muspel, we want to solve several existing problems. First, there is the dificulty to interconnect and communicate different platforms, smart objects and existing sensors. Nowadays, because of the many different software and hardware, it's necessary to have some technical knowledge about each element, and a specific application for each mechanism. The problem we want to solve is how to make this easy for any kind of user, because all this requires a very technical knowledge and so it's limited to just a small percentage of people. That's why, by using a framework, we try to make the interconnection of heterogeneous smart objects totally accesible for any user who knows the smart objects.
An example is the creation of a sensor net and its configuration in a graphic form. It is as simple as dragging boxes and writing the graphics we need and we ask for. By doing this, any person, regardless of their computer skills, could create an application based in a minimum knowledge of control.
This makes the same application a good one for modeling, too, without having to worry about different kinds of mechanism connections, because the user would not have to take into account if the mechanism is Bluetooth, Wireless, infrared, Wi-Fi or HTTP.
All things created with this model are abstracted out of all short and medium configuration processes. In this way, the user -who only knows about the model control- can create sensors and different applications which administer different actions according to the user's condition.

CONTRIBUTION

With this research we intend to make the following contributions:
Simplicity of software: it will be offered in a easy and intuitive form, exact or graphic, like the user wants, to create the applications. In this way, the user will only have to know each element (sensor, action, smart object…) and the different configurations he can apply to them, that is, if it is allowed. This way, any user, even without having knowledge in programming or electronics, could create and register an application based in sensors.
Reutilization of used technologies: A lot of programs that use the same software can be done.
Creation of an abstraction layer in a bigger level: to configurate and work with different connection kinds in a low level is something to avoid, for example Bluetooth, Wi-Fi, Wireless, SMS, HTTP,… All this will be done clearly and the user will only have to choose the connection that he wants to perform with the object, if this one can be accessed through several connections.
Domain Specific Language for creating of applications: A domain specific language will be created to serve as a base for creating applications based in sensors.
Domain Specific Language for exchange of data: A domain specific language will be created to serve as a base for exchanging data between smart objects.

Metamodel: A specific model that serve will be created as a base for creating some models for applications based in sensors and which want to solve similar problems.
Data centralization: A system that will be able to centralize the developed applications will be created. This way, any user will be able to get access to public content published by other people.
Connection abstraction: Thanks to the application, we will abstract and encapsulate all connections and communications of objects to Internet. This is because all objects that wants to send and receive data might connect or perform a necessary request to the application. This way it won't be necessary to use other protocols that can be slower, spend less battery or aren't available.

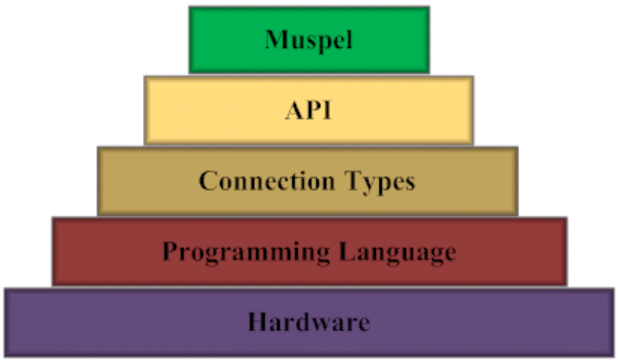

Figure 1. Muspel's Application Layers

The Figure 1 shows that our contribution is an abstraction over API offered by different systems, connection kinds, programming languages and the Hardware itself.

STATE OF THE ART

Smart cities are the pinnacle of sensor nets. In them the situation of certain zones is watched and also certain mechanisms act accordingly.
The most classical example are the lights. These ones turn on when the sensor doesn't receive a minimum amount of light previously fixed.
By implementing this system we could make possible for the smart city to send a text message to the subscriber's cellular and inform about a certain situation. For example, the traffic on the road. This example can be extended to parkings where we are notified about free places even before entering.
These configurations are created by the government or specific people thanks to a certain software. All this is not accessible for users who cannot add a sensor to the government's net.
However, there are certain web sites that offer users the possibility of sharing the measurings done by their own sponsors. This is the case of COSM (previously known Pachube) (LogMeIn, 2013). Other examples are Paraimpu (Piras, Carboni, Pintus, & Features, 2012) o ThingSpeak (IoBridge, 2013), etc.
COSM offers its API and several advices to work over the mother base Arduino. Despite this, the user will have to learn how to use the API and work in a lower lever with Arduino, because it is programmed in a programming language C. It offers data in different forms: XML, JSON, CSV. We can access to them by a free and bidirectionnal API RESTful (upload and download of data) available in several programming languages (C, Ruby, PHP, JavaScript, …),as in modifiable graphics in the web. It allows users to upload their applications in a public or private way and a total administration of them and their data. It also allows users to chat in the same page where the sensors are.
ThingSpeak offers the same than COSM, but there is a diference. It offers an only API for connecting to the service and it doesn't give API to work with any system, like Arduino.
There are pages and a lot of different mechanisms to interact according to conditions. Among them, there are some which are intelligent. The most used nowadays is Smartphone. But Smart TVs, tablets, NFC, RFID and also the laptops are getting more presence at homes, in the streets or at establishments. It can interact with each one in several ways: Wi-Fi, Wireless, infrared, Bluetooth, SMS, call, MMS, HTTP,

SMTP, IMAP,… The only problem is connecting with them. Sometimes it is simple, for example when we send an electronic message. Other times it can be more complex, as sending a SMS, MMS or connecting Bluetooth; this is due to the absence of information, examples or the system complexity. Because of these problems, It is needed to know the sphere and the different programming languages, technologies and mechanisms. A lot of knowledge is required just to do a few things.
Because of that, with the introduction of the Model-driven engineering we intend to make all this process easier for the user thanks to the introduction of an abstraction layer. Users will have to know the sphere and have a little knowledge about each tool. However, this last process will be drastically simplified to offer a very easy system for creating sensor nets next to conditions that are necessary in a certain act.
Our goal is to create a web site that centralizes all the information sent by users as well as the register of its applications. We will provide a RESTful IAP, too, so a download data can be sent from different notations. We will also provide a framework to make possible to create applications in an easy way for different systems, like Smartphones with sensors and sensor base plates. By doing this, the user, just by using this framework, would be able to create a native application that uses the Smartphone's sensors or base plate and send the desired information to the web. To send all data a standard DSL will be created, one that will be compatible with any application or any sensor. With all this we intend to make the extension through new systems and sensors quicker and easier.

DESCRIPTION

Muspel

Muspel will be established from the web site and from different APIs and frameworks. The web site will centralize all user information and his application's register. It will be responsible for managing this information and, via RESTful service, providing the necessary data required by the applications.
To make the interaction easier for the user, we will put a set of APIs with which he could connect to the service and some frameworks which could be used to develop the application in an easier way, from Smartphone to base plates.
All the available information would be accessible in different forms, like XML, JSON y CSV. In spite of that, for direct communication whit applications, it would be necessary to create a standard DSL that could serve to send data. This way, the future extension to more systems and sensors would be easier.

System & equipment

The main piece for connecting sensors to the environment will be a mother base Arduino (Hribernik, Ghrairi, Hans, & Thoben, 2011). The required sensors of data and several mechanisms are used to perform actions like loudspeakers, LED's, motors, servomotors… These will be connected to the mother base Arduino (Yamanoue, Oda, & Shimozono, 2012).
The Arduino will be connected to the PC via USB on the first tests: then, through a Wireless unit (Georgitzikis, Akribopoulos, & Chatzigiannakis, 2012). For the performing of all the tests there will be an exclusive computer that will process and save all the data, and it will make the connection with different systems and smart objects.
Several Smartphones will be used, each one with different operating systems like smart objects, sensors, and text messages, multimedia messages, calls or mails according to the configuration.
Other kind of smart objects which will be used are Smart TVs, sensor nets, tablets and computers.

Software

Like IDE we will be using Eclipse. For doing the metamodel and the text and graphic environment we will use Eclipse Modeling Framework (Hegedus, Horvath, Rath, Ujhelyi, & Varro, 2011), Graphical Modeling Framework (Kolovos et al., 2009) and Textual Modeling Framework. This way we get a development almost for free.

We will use the Java program language; that will create an application which will be easily ported to different known systems. The only requirement is to install the Java virtual machine so it can work with the application.
To create the model it an Ecore metamodel will be used.
Herewith it wants to create a framework to implement applications in a simple way in cellular phones or well to use frameworks of the graphic or textual interface. If the first one is used, a user with programming knowledge should be able to create a new design and implement it. This way, he could use it in a lower level and make a deeper configuration of the application. With the last two, any user without programming knowledge could implement an application because the domain is the only thing he needs to know. With the framework of graphic interface, the user only might drag and drop boxes and fill his data to generate the application. By using textual framework, the user would have to write, according to some rules which would be given in real time, the application in each text.

Architecture

Now we will describe the internal architecture and the operations of the web service, as well as the way in which sensors and devices interact with it.

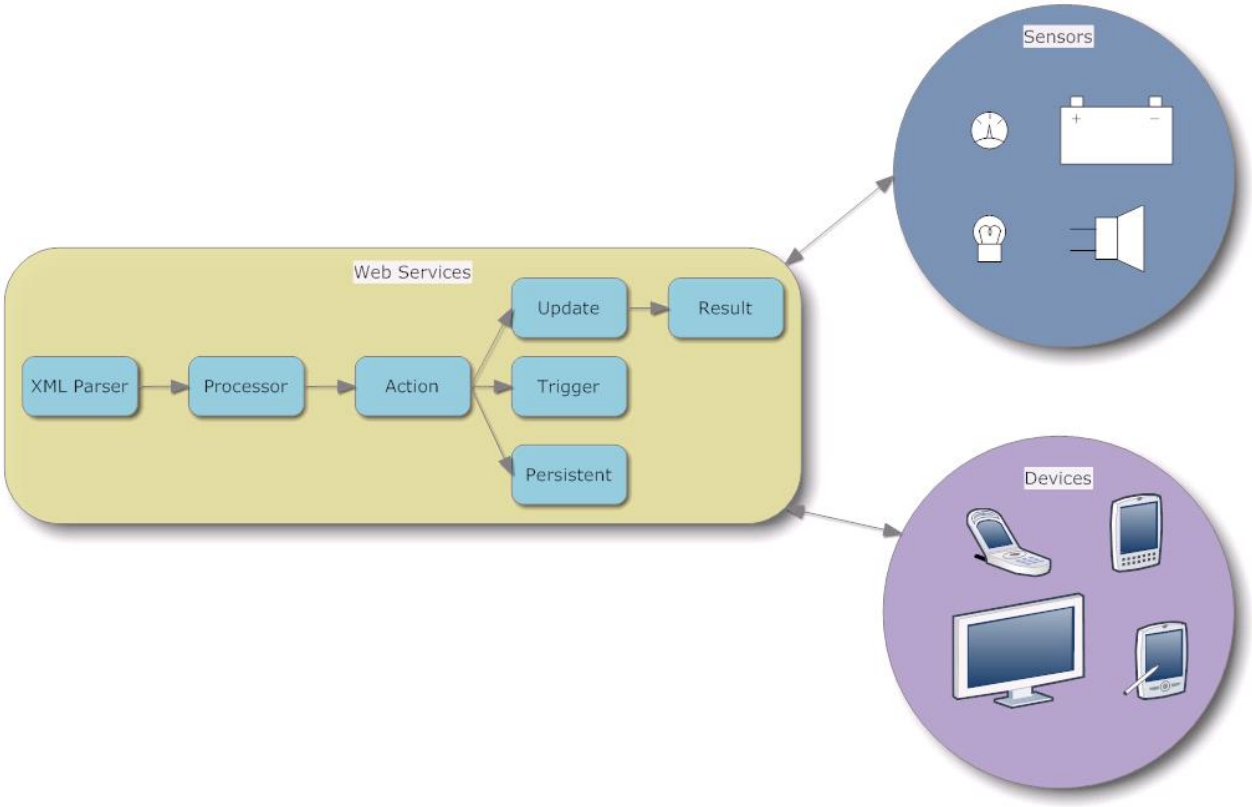

Figure 2. Internal Architecture of application web service in altogether with sensors and devices.

First of all, to explain that the way to interact with the web service that is sending the text in XML format, we will describe the internal operations of the web service. This has a series of steps:
When a XML file is received, it processes that one en the XML Parser platform. Once it processes the file, it sends the data to the processor. This one, from this data, knows if it is a sending request of data or an order, and then it runs a series of actions: Update, Trigger and Persistent. All three will be run at the same time.

If the persistence configuration is activated, one of these actions will be kept (or not) within the data base. If the processed action must break out a process, the Trigger will be the one which makes it. For example, when a temperature sensor shows 100ºC, it will send a call to the firefighters and an e-mail to the cellular of the sensor's owner.
At last, the Update executes the necessary update in the web service and in the respective URL; it will always keep the RESTful service. It creates, erases or updates new files when necessary; also, new URL, images or folders…
Once the last process I finished, the final result will be returned. This can be a XML with successful confirmation of the processing of the process trigger. If data or one or more text files are sent with the order, according to the asked format, as well as the respective XML to send the required orders.

Applications

First we will create the Ecore with the Eclipse Modeling Framework and after that, the graphic editor and the text editor will be created thanks to the Graphical Modeling Framework and Textual Modeling Framework.

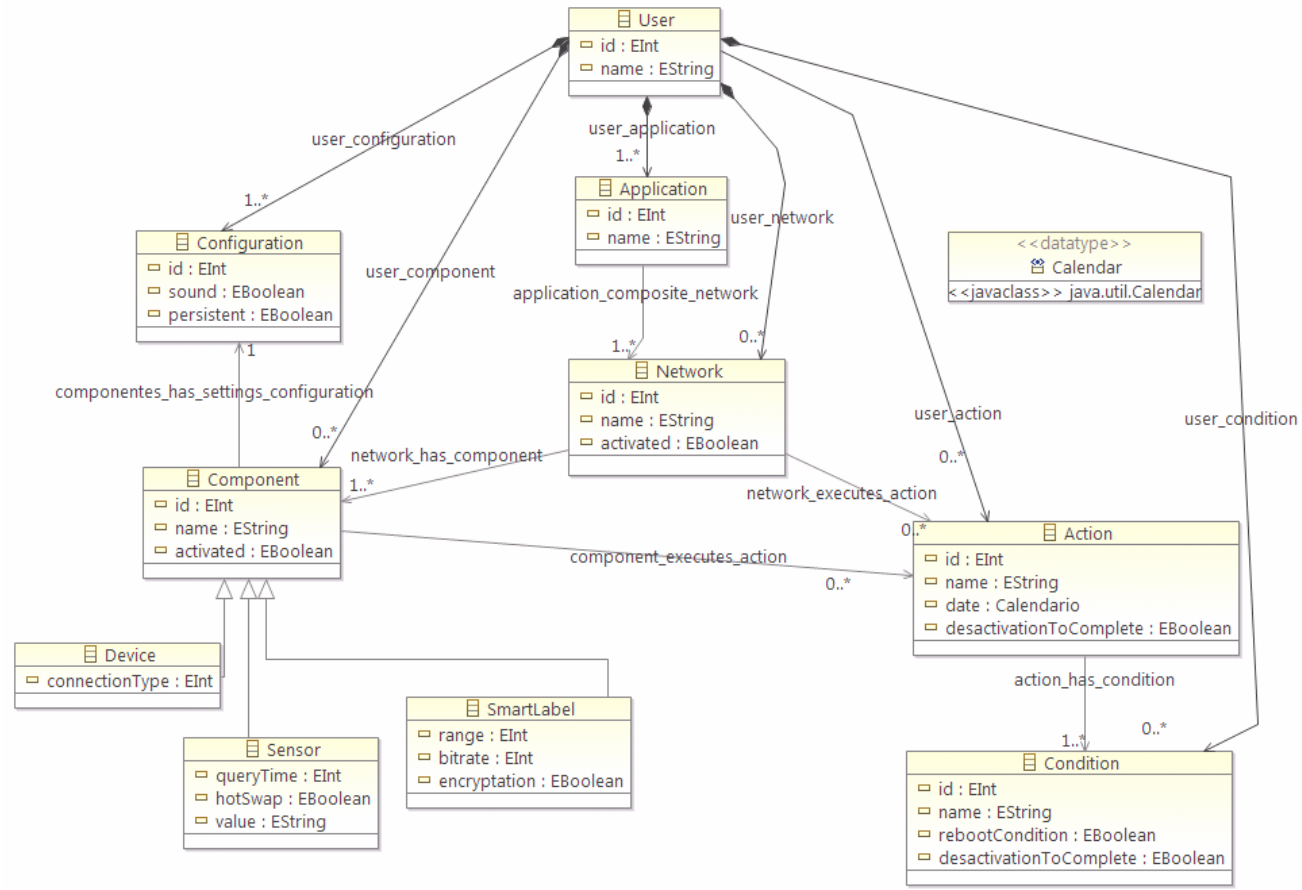

Figure 3. Ecore metamodel used to describe elements, connections and limitations of modeling language that will be used.

After that, we need to create all the elements required to abstract whatever we want through the model. That is, different kinds of connection, options, configurations, etc.
In other way, its temperature, humidity, movement and light sensors will be connected to the mother base Arduino; as well as mechanisms like loudspeakers, LED's, a motor, a servomotor and a relay.

Next step is programming sensors so they can receive the wanted values at the time we want.
After that, the Java model has to be modified to be connected to Arduino and, this way, the configuration that was created as a model interacts directly with the hardware, so the data can be processed in both directions: from model to Arduino and vice versa.

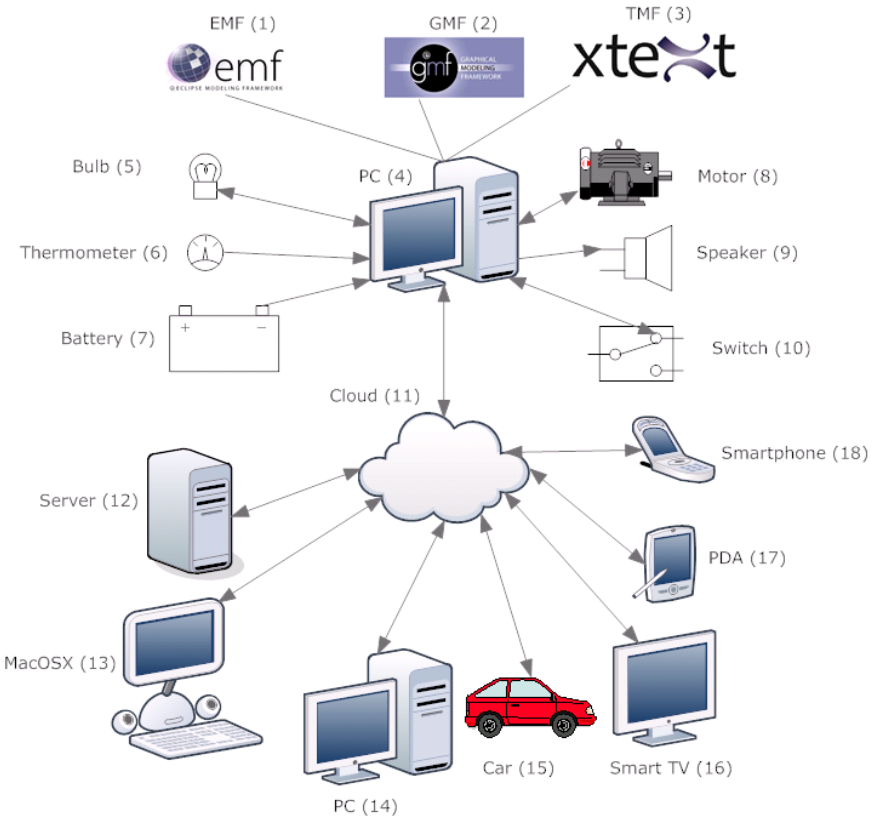

Figure 4. Example of system architecture where it is shown the interconnection of different elements: from sensor interact, going through the cloud, to arrive to smart objects.

In Figure 4 We can see an illustrated example of the prototype. Through frameworks of Eclipse, the model is created. To create the Ecore metamodel we use the Eclipse Modeling Framework (1). To create the graphic framework we use the Graphical Modeling Framework (2) and to create the textual framework, the Textual Modeling Framework (3). All that is in a PC that we use to read different systems (4): In this case, the PC (4) has connected 6 mechanisms.

- Bulb (5): It has bidirectional connection. We can know if it is switched on or switched off and also turn it on or turn it off. Because of that we can, under certain conditions, turn it on or turn it off or know if we need it to be in the state in which it is at that moment.
- Thermometer (6): We can only check its state, that is, the temperature that it gives us. This one can be located at home, outside or in a mobile mechanism.
- Battery (7): We can only check its state, that is, the remaining battery. For example, in a cellular phone or a laptop.
- Motor (8): It has a bidirectional connection. We can know if it is switched on or switched off and turn it on or turn it off under certain conditions or by direct orders from an IOT mechanism. It can be the load motor of a compressor and, if it is detected that is full, it can stop or inform us and, this way, alter the power that is given to the battery load.

- Speaker (9): We only can give orders. In this case, it can be used to alert us of certain states. For example, when the battery is over 5% or when the motor does not work, it sends an alarm warning.
- Switch (10): Through its bidirectional communication we can know the state of this element and control it. For example, if this belongs to the heating, we can turn it on or turn it off depending on the exterior temperature and, even from the Smartphone (18) or the PDA (17), send an order to turn it off until the home thermometer has reached 21ºC.
- Cloud (11): The cloud is Internet. It is a resource which connects all mechanisms, all the data is sent through it, and it is also the calls to other mechanism are performed.
- Server (12): Server that can receive and send data. This one can be used to keep mechanism data and send e-mails to different mechanisms when they meet a certain condition, like the compressor and the battery being full and the temperature surpassing 45ºC.
- Mac OS X (13): A computer with an OS X operative system that can interact in a bidirectional way with the cloud, that is, with different mechanism connected to it.
- PC (14): A computer with a Microsoft Windows or a GNU/Linux operative system that can interact in the same way as the Mac OS X.
- Car (15): Car with bidirectional connection, that car transmits its GPS position in a public or private form, even other data and emergency signals. Also, it can receive notifications of other IOT mechanisms. For example, from another car that has had an accident and it is within 1 KM.
- Smart TV (16): This device has a bidirectional connection. Thanks to it, video calls or VOIP calls can be sent and received.
- PDA (17): By PDA the data can be sent and received to operate with different mechanisms. For example, if we notice that our house temperature sensor shows 10ºC, we can send an order to turn on the heat.
- Smartphone (18): As the same way than PDA (17), we can interact with other smart devices, as well as receiving the GPS position of the car (15) in the case of an accident.

EXPERIMENTS

Prototype: Design of an application for interconnection and its use in a sensor base.

In this section, we will describe how the proposed platform is used to define an application that adds an application created by a user. For this example, we have an application with tow sensors (humidity and temperature), a Smartphone and a NFC label. Everyone will share the same configuration and will perform the same actions. There are two conditions: a temperature beyond 40º and a humidity level below 60%. That's when it will be necessary to call the firemen.

That application is composed of a Network that has got a Smartphone from which we use the temperature and humidity sensors and a Smart Label as NFC. The Smartphone has a configuration where sounds and persistence are allowed. The action created for Smartphone, sensors, the NFC Label calls the firefighters if it happens at 13/12/2012 and the temperature is over 40º C and humidity is less than 60%. The first condition, once it's done, does not begin again. Even if the temperature decreases, this condition will not be taken into account. The opposite case is the humidity, because, for the action jump, it will have to be less than 60%. If the humidity condition is met first, or it happens that the temperature is over 40º and the humidity is over 60%, the action won't be executed until the humidity levels are below 60% again, the action won't jump until the humidity goes back to normal.

The next figure shows an example of the application by the text editor. First it is necessary to write the user's name; then the information that will follow this order:

1. One or more applications.
2. One or more networks with ID attributed in the application.

3. One or more components with the ID attributed in networks. All these can be devices (Smartphones, speakers…), sensors, Smart Labels (RFID, NFC).
4. Configuration of components.
5. Their actions.
6. Conditions for the actions that are carried out

```
User id: UO206639
Name: "Cristian"
Application {
    Id: app1
    Name: "Muspel"
    Networks: network1
    }
Network {
    Id: network1
    Name: "Muspel One"
    Activated: TRUE
    Components: device1 sensorTemp sensorHum sl4
    Actions: actionCallFirefighters
    }
Device {
    Id: device1
    Name: "Smartphone 1"
    Activated: TRUE
    Connection Type: "SMS"
    Configuration: conf1
    Actions: actionCallFirefighters
    }
Sensor {
    Id: sensorTemp
    Name: "Temperature"
    Activated: TRUE
    Query Time: 10000
    Hot Swap: TRUE
    Configuration: conf1
    Actions: actionCallFirefighters
    }
Sensor {
    Id: sensorHum
    Name: "Humidity"
    Activated: TRUE
    Query Time: 20000
    Hot Swap: FALSE
    Configuration: conf1
    Actions: actionCallFirefighters
    }
SmartLabel {
    Id: sl4
    Name: "NFC Active"
    Activated: TRUE
    Range: 50
    Bitrate: 30
    Encryptation: FALSE
    Configuration: conf1
    Actions: actionCallFirefighters
    }
Configuration {
    Id: conf1
    sound: TRUE
    persistent: TRUE
    }
Action {
    Id: actionCallFirefighters
    Name: "Call firefighters"
    Date (dd/mm/yy): "13/12/12"
    Desactivation to complete: TRUE
    Conditions: conditionTemp40 conditionHum60
    }
Condition {
    Id: conditionTemp40
    Name: "Temperature > 40"
    Reboot Condition: FALSE
    Desactivation to complete: TRUE
    }
Condition {
    Id: conditionHum60
    Name: "Humidity < 60"
    Reboot Condition: TRUE
    Desactivation to complete: FALSE
    }
```

Figure 4. Code snippet of an application made with the textual editor.

In next figure it can be seen the same example, but presented with the graphic framework. In it we can see that the systems are connected by lines and each one contains its information.

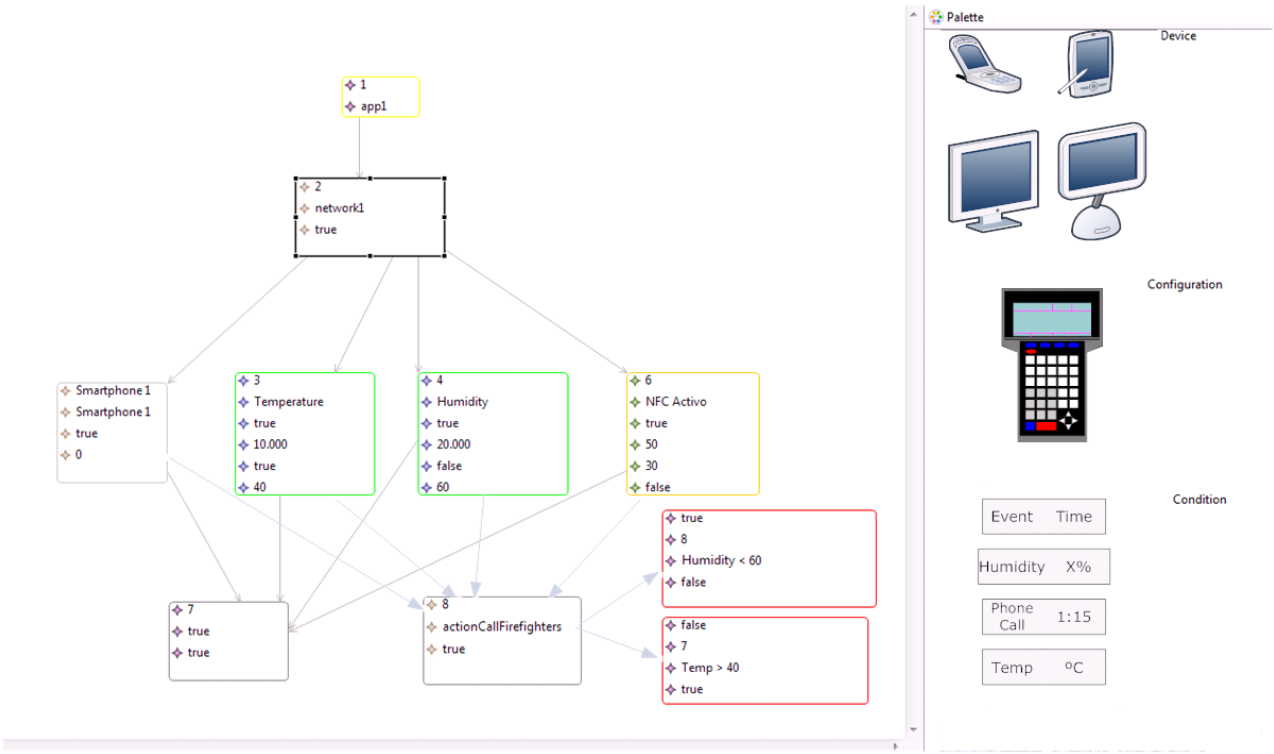

Figure 5. The same application made with de graphic editor.

In the figure 5 we can see the same application than the ones in the example of Figure 4. The identifier 1 matches with the application. The identifier 2 is the Network that has got 4 devices. The Smartphone 1, the 3 is the temperature sensor, the 4 is the humidity sensor and the 6 is the Smart Label as NFC. These four devices have the same configuration, one that matches with identifier 7. They have the same actions, identifier 8. This has two conditions: humidity under 60% and temperature over 40ºC.

FUTURE WORK

Future work that could be done would be:

- The mobile mechanisms that are connected to the platform need a framework which makes the synchronization and communication with the platform. To make possible for several electronic mechanisms with different features and different operative systems to connect to the platform it's required that this connecting application is implemented in several programming languages; this way the application that adapts better to the specifications could be installed in each mechanism. Because of that, part of the future work could be to create native frameworks in other platforms and in other languages, like: C++, C#, Ruby, PHP, JavaScript, Objective-C, …
- To complete the specific language capacities of graphic domain to allow the inexperienced users to specify business processes which involve the combined work of several smart objects and mechanisms. To develop this new specific language of graphic domain we want to evaluate some languages and tools designed to model business processes.
- Thanks to the heterogeneous nature of mechanisms with its own operative systems and its different functioning, it would be interesting to expand to a bigger number of systems: Smartphones (Ubuntu, Windows phone, Firefox OS), video consoles (PS3, Wii, Wii U, Xbox 360, Nintendo DS, Nintendo 3DS), computers (Microsoft Windows, Mac OS X, GNU/Linux), graphic card with sensors (NVidia, ATI), Smarts Cities…

- Actually, every mechanism has several forms of interconnection with the exterior. For example, a Smartphone or cellular phone with infrared, Bluetooth, *Short* Message Service (SMS), Multimedia Messaging System (MMS) and Wireless. A computer can have some of these and even connection ports, like Universal Serial Bus (USB), LPT1 ports o Ethernet ports. Other devices have High Speed Downlink Packet Access (HSDPA), (High-Speed Uplink Packet Access (HSUPA), Global System for Mobile communications (GSM), Worldwide Interoperability for Microwave Access (WiMax), Long Term Evolution (LTE), Universal Mobile Telecommunications System (UMTS), Code Division Multiple Access (CDMA),… Because of this many possible forms of data sending, it can be increased to give support to a lot of protocols and give more possibilities in the interaction with other mechanisms or serve as an extra support if there is a failure in one of the used objects.

CONCLUSIONS

As it has been demonstrated, the way to abstract the configuration, organization and creation based in sensors has been taken to a very high level thanks to the use of model-driven engineering.
Users can choose between a textual environment that drive it or a graphic environment where they drop the boxes, fill them with data and joint them.
Among data, user only has to choose those one he prefers, choose the kind of connection, conditions, actions, persistence, and he won't have to take care of how it operates in the inside.
All that makes possible for unconnected people (as long as they know the sphere) to create applications of sensor nets for personal use in an easy and quicker way, which can be used and modified again. They can create a net at home or at work, or one net in each room and they can modify it in a little time. In the primary sector, they can create sensor nets to control fields, greenhouses, and automate tasks in a simply way, according to a mechanism that receives the action, for example, a motor or a robot.

REFERENCES

Falvo, M. C., Lamedica, R., & Ruvio, A. (2012). An environmental sustainable transport system: A trolley-buses Line for Cosenza city. *International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion*, 1479–1485. Ieee.

Georgitzikis, V., Akribopoulos, O., & Chatzigiannakis, I. (2012). Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks, *10*(3), 1686–1689.

Gu, H., & Wang, D. (2009). A Content-aware Fridge Based on RFID in Smart Home for Home-Healthcare, *02*, 987–990.

Hao, C., Lei, X., & Yan, Z. (2012). The application and Implementation research of Smart City in China, (70172014), 288–292.

Hegedus, A., Horvath, A., Rath, I., Ujhelyi, Z., & Varro, D. (2011). Implementing efficient Model Validation in EMF Tools.

Hribernik, K. A., Ghrairi, Z., Hans, C., & Thoben, K. (2011). Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, (Ice), 1–9.

IoBridge. (2013). Thingspeak. Retrieved from http://www.thingspeak.com

Kolovos, D. S., Rose, L. M., Paige, R. F., & Polack, F. a. C. (2009). Raising the level of abstraction in the development of GMF-based graphical model editors. *2009 ICSE Workshop on Modeling in Software Engineering*, 13–19. Ieee.

LogMeIn. (2013). COSM. Retrieved January 15, 2013, from https://cosm.com/

Piras, A., Carboni, D., Pintus, A., & Features, D. M. T. (2012). A Platform to Collect , Manage and Share Heterogeneous Sensor Data, 1–2.

Rothensee, M. (2007). A high-fidelity simulation of the smart fridge enabling product-based services. *3rd IET International Conference on Intelligent Environments (IE 07), 2007*, 529–532. Iee.

Vienna, U. of. (2013). European Smart Cities. Retrieved November 26, 2012, from http://www.smart-cities.eu

Yamanoue, T., Oda, K., & Shimozono, K. (2012). A M2M System Using Arduino, Android and Wiki Software. *2012 IIAI International Conference on Advanced Applied Informatics*, 123–128. Ieee. Retrieved January 13, 2013, from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6337170

KEY TERMS AND DEFINITIONS

Internet of Things: network of daily objects connected to each other to allow an interaction among them.
Model-driven engineering: methodology of software development that is focused on model creation near the concept of private domain instead of software.
Sensor: mechanism that is capable of detecting physic or chemical magnitudes. For example: temperature, light, humidity, CO_2, radiation…
Sensor platforms: Set of sensors in a same platform to allow the process of interconnection and make it easier.
Sensor-based applications: applications created to work and interact with sensors.
Smart objects: objects that can interact with others. For example, two Smartphones or a Smartphone with the TV.
Domain Specific Language: language given to solve or involve a specific problem and provide a technique for solving private situations.
Web Services: technology that uses a series of protocols and standards to exchange data between applications.
Smart City: City that creates an environment where different mechanisms are interconnects among them, all by using sensors and automations.