Computer Networks 64 (2014) 143–158

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios

Cristian González García *, B. Cristina Pelayo G-Bustelo, Jordán Pascual Espada, Guillermo Cueva-Fernandez

University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain

ARTICLE INFO

Article history:
Received 3 July 2013
Received in revised form 8 January 2014
Accepted 7 February 2014
Available online 17 February 2014

Keywords:
Internet of Things
Ubiquitous computing
Sensor network
Model Driven Engineering
Domain Specific Language
Smart Objects

ABSTRACT

Smart Objects and Internet of Things are two ideas that describe the future. The interconnection of objects can make them intelligent or expand their intelligence. This is achieved by a network that connects all the objects in the world. A network where most of the data traffic comes from objects instead of people. Cities, houses, cars or any other objects that come to life, respond, work and make their owner's life easier. This is part of that future. But first, there are many basic problems that must be solved. In this paper we propose solutions for many of these problems: the interconnection of ubiquitous, heterogeneous objects and the generation of applications allow inexperienced people to interconnect them. For that purpose, we present three possible solutions: a Domain Specific Language capable of abstracting the application generation problem; a graphic editor that simplifies the creation of that DSL; and an IoT platform (Midgar) able to interconnect different objects between them. Through Midgar, you can register objects and create interconnection between ubiquitous and heterogeneous objects through a graphic editor that generates a model defined by the DSL. From this model, Midgar generates the interconnection defined by the user with the graphical editor.

1. Introduction

Internet of Things (IoT) opens the door to many possibilities, like the field of environmental intelligence [1], better known as Smart Earth [2], the Smart Cities [2] or the Smart Homes [1,3,4]. This is due to a recent rise in the use of new technologies, like computers, the Internet, cloud computing, Smartphones [5], tablets, micro-controllers [6,7] and intelligent tags [8,9]. Also, the price of sensors are now much lower than before and wireless connections have been quite improved [10]. This entire set of objects explains the relevance that the Internet of Things is gaining in telecommunication [11].

IoT allows us to place dispersed sensors and objects in order to receive information from any part in the world. These may be located in our house, in an accessible place or even inside a machine or a motor. Others might probably be located in the depths of the ocean to detect meteorological phenomena. Even inside a nuclear plant or in a sealed room full of chemical substances. Maybe behind the enemy lines in a battlefield or even in places where they cannot be reached after being set [10,12]. With this we can establish an interconnection between those heterogeneous objects

* Corresponding author. Tel.: +34 985103397.
E-mail addresses: gonzalezgarciacristian@hotmail.com (C. González García), crispelayo@uniovi.es (B.C. Pelayo G-Bustelo), pascualjordan@uniovi.es (J. Pascual Espada), cuevafdz@gmail.com (G. Cueva-Fernandez).

http://dx.doi.org/10.1016/j.comnet.2014.02.010
1389-1286/

as long as they have access to the Internet [13]. These objects can be sensors dispersed through the terrain [10], a computer [14], a net of sensors [10], an Arduino micro-controller, a Smart TV, a Smartphone, a Radio Frequency ID (RFID) intelligent tag or a freezer. Internet of Things is the interconnection of heterogeneous objects through the Internet [3,9,13,15,16]. This will lead to a future in where it will not be used just to communicate people, but also to communicate people with machines and even machines with machines (M2M) [8]. Thanks to this, Internet of Things will become a great impact in our daily routines, both for personal and commercial use [11]. It can improve our quality of life and the enterprise production. All through the interconnection of objects and all the possibilities that it offers.

It must be highlighted that Internet of Things is gaining so much importance that even the United States National Intelligence Council considers it as one of the six technologies that will have more weight in the interests of the United States from here to 2025 [17]. On the other hand, back in 2005, at Tunisia, the ONU predicted a new era of ubiquity where the Internet traffic will be smaller in comparison to the traffic generated by objects from our daily routines [11].

That is why Smart Objects are also important: these devices can interact with others thanks to certain actions or events [3]. An example of this can be found in Smartphones, micro-controllers like Arduino [3,6,7] or any other object capable of managing information [18]. When combined with the Internet of Things, we obtain a network that not only transfers data, but also gives it intelligence and actions according to the information captured by these objects.

For instance, some proposals [1,19] present a freezer that analyses the habits of alimentation. Potentially it could be very useful for people with illnesses or even to improve the quality of life of families. In the work found in [3], K.A. Hribernik investigated an intelligent system to give a certain amount of intelligence to a house. Additionally it could control a forklift using certain conditions activated by its different sensors. This can also improve ecology and economy, because it can save energy in a campus by using RFID intelligent tags, as [8] proposed, but this could also be applied to a house [13].

A more ambitious example is the connection of sensors for environmental monitoring. This could be critical to prevent or control certain kinds of circumstances that could play an important role [20]. For instance, in the disaster of the Deepwater Horizon in 2010 at Gulf of Mexico it could have been used to control the spill of oil, analyse the amount of salt in water and the status of different areas to prevent bigger problems [20]. Moreover, IoT could prevent many disasters by monitoring different structure conditions.

Due to the above, the purpose of this investigation is to design a Domain Specific Language (DSL) in addition to the implementation of a graphic editor. The editor will allow us to create applications for interconnected heterogeneous objects: sensors, Smart Objects, etc. We intend to allow anyone to interconnect many objects without any programming knowledge. Thanks to our DSL, we offer the user an abstraction of the base programming language, so users will just have to know the domain of the problem and its properties. The editor will make the writing process easier for them. Also, it implies the creation of an infrastructure to support the communication between the different objects. The platform will take care of the centralization of the data of each sensor and object, allowing them to interact and communicate between them through the creation of a ubiquitous network by solving the interconnection problem.

The remainder of this paper is structured as following: In section 2, there is an introduction to the currently existing problems in IoT. In section 3 we discuss the topic of Internet of Things, Smart Objects, Model Driven Engineering and the DSL. Furthermore, we present out research on the existing IoT platforms. Section 4 shows the case study of this research in detail: the Midgar IoT platform. In section 5 we cover the evaluation and discussion of the data obtained from participant surveys. Section 6 contains the conclusions of this paper. Finally, in Section 7 we describe many possible options for future work that can be considered from the results of this research.

2. Challenges of the Internet of Things

With the Internet of Things and the Smart Objects we can create a Smart Network to control almost anything. However, there is a big common problem. The heterogeneity of objects does not allow them to interact with each other [21]. To connect them, we need to develop a common interface, such as an application hosted in both of them that communicates with a server. We can see an example of this in existing multiplatform programs, from messaging services like WhatsApp, Skype or Line, to multiplayer videogames like Atriviate and Angry Words. However, these applications can only connect certain predefined Smart Objects and can only be developed by people with knowledge in software development. IoT, however, offers a great support for heterogeneous objects, without taking into consideration the operating system, platform or their functionalities. Likewise, it does not matter if the objects are intelligent or if they are domestic objects or sensors scattered throughout the world.

Individuals without programming knowledge want to connect objects to take advantage of the technology to improve their environment. The general public should only comprehend the domain of the problem without worrying about the software development. Our solution is to create a Domain Specific Language to reach that abstraction layer and encapsulate the domain of the problem in it [22].

2.1. Heterogeneity of objects

One of the main problems of the Internet of Things and sensor networks are quite similar due to the nature and purpose they share. Both are composed of objects that must be connected, but in the second case, there are only sensors. At this point, the problem consists on the different implementations that must be done for the different operating systems. From the software point of view, what can differ in each case is the type of message that is being sent.

That is why messages sent between Smart Objects created by an enterprise can be very different from the ones sent by the Smart Objects of another company. This makes impossible for Smart Objects of different enterprises to interact, as there is no understanding between them [21]. The solution for these problems would consist in using a network which could connect different devices [8]. This can process all the received data and corroborate everything under the same format and give the necessary intelligence and understanding to each device through the network. The main problem is the heterogeneity, the dynamism and evolution of its content [21]. From the three main problems that the content of the network can cause, the one that matters for this research is the one about heterogeneity.

The problems arise from the intention of managing different devices, protocols, sensors, objects and applications. Each one developed in a different way, by different manufacturers and with barely any common elements. There is not a standardisation that encompasses all of these objects in every aspect (protocols, available services, types of data or message sending). This implies that direct communication will not be always possible, due to the lack of understanding and the absence of interfaces or a standard protocol [23]. As an exception, we should mention Bluetooth and infrared sensors. This difference between objects is what we call object heterogeneity. That is why we must provide an adequate architecture that gives support to an easy communication with any type of device. An example of heterogeneity with a Smart Object can be an Arduino micro-controller [3,6]. In this case, the heterogeneity of sensors is bigger because of the great amount of sensors that can be connected to the microcontroller. Any device of any brand can be connected to this one. That implies that any Arduino can be different from the rest depending on the sensors that are used and the data they capture. Another example of heterogeneity problems (on a bigger scale) in Internet of Things is Smart Cities. Has [2] argues, the first problem that must be solved is the recollection of data, the access and the transmission. This is why the Internet of Things needs to recognize Smart Objects and keep a constant flow of messages between those objects. However, for each area in which a sensor network is established, each researcher solves the problem in a different way [10]. That implies multiple solutions but none of them can become a standard [23].

The solution is an architecture that supports the sending of messages coming from the different types of devices and it is able to respond to each one. Therefore, we will create a service-oriented architecture (SOA), as proposed [11,21,24,25] and a RESTful application that can be used by any object, as proposed in [16,26].

2.2. Development of applications to interconnect objects for end-users

The development of applications requires people with experience in software development, as well as knowledge of the different programming languages that are needed to use in each platform. Also they must have a wide knowledge of the domain of sensors and objects to interconnect. The main problem is to make possible for people without software development knowledge to create these applications in a quick and simple way [23]. The solution is to offer an application to help them with this task. This way, if people are willing to do so, they will be able to interconnect the different devices and sensors that implement the developed specification and thus perform actions under the circumstances of their choice. As a solution, we will propose the creation of a graphic editor that makes easier to develop that application using a Domain Specific Language (DSL). That is how, through this DSL, we will abstract the development language, allowing inexperienced users to define it by using tags and attributes without having to program at all.

3. State of the art

In the context of the Internet of Things, different Smart Objects interact with each other, usually through the exchange of messages. An object performing an action when another, located thousands of kilometers away, transmits certain relevant information. An intelligent network capable to take decisions based upon certain actions. The following is the definition of Internet of Things: one network with many of these objects, Smart Objects and sensors. With this set of elements, any person will be able to automate and improve his daily life. Cities will become alive. However, there are two big problems that must be solved: First, the interconnection of heterogeneous objects, in order to work together within the same network, under the same protocol or standard [23]. Second, not everyone is able to program. Both problems could be solved by themselves. The first one could be solved establishing a common standard for all the creators of objects. The second, teaching everyone how to program. However, none of these solutions is viable.

Therefore, a possible solution would consist in allowing people to create applications for interconnecting objects without having to program, only needing to know the domain of the problem. To reach this goal, we need to abstract the problem as much as we can. Through Model Driven Engineering (MDE) and a Domain Specific Language (DSL) it is possible to prevent people from having to program. We only need a language that encapsulates this domain.

3.1. Internet of Things

Nowadays, most of the interactions performed through the Internet consist on the human–human connections. However, everyday more and more Smart Objects can connect themselves to the Internet and in the future we can expect to have more connected Smart Objects than persons [8]. These objects can interact between themselves, send data to each other and perform certain action when certain conditions are met. Heterogeneous objects interacting between themselves: That is the definition of Internet of Things [3,8,9,11,13,21]. IoT appeared in response to the necessities of the supply chains and the identification of objects, persons and animal through the

use of RFID intelligent tags [9,11,21]. Subsequently, with the use of RFID, we can assign a single identifier to an object [11]. We can locate a certain object for a certain task. However, as [8] explains, for the Internet of Things to exist, three steps are required: integrated intelligence, connectivity and interaction. That is why the base for Internet of Things are the sensors and the RFID cards [8,11]. And yet, these are not the only ones. According to [11], Near Field Communications (NFC), the sensor networks, Actuators Networks (WSAN) and RFID, "*are the atomic components that will merge the real world with the digital world*". Sensors can capture almost any kind of information, change or data. These can be processed by a Smart Object (a Smartphone, for instance) or sent to a server. By using the RFID intelligent tags we can identify different objects [21], so, through radiofrequency, we can know the properties of this element and monitor it [9]. If we combine both of them, a certain object will be able to perform an action as a reaction to an event captured by a certain sensor. To achieve these objectives, we need an intelligent infrastructure capable of connecting the different objects and granting them enough intelligence to interact with each other. In some cases communicating them the decision they have to make [8]. It would only be necessary if the Smart Objects do not have intelligence of their own or if they need it in a certain moment because of an external decision.

3.2. Smart Objects

As [3,11,27,28] explain, a Smart Object, also known as Intelligent Product, is a physical element with different properties, identifiable through its useful life, that interacts with the environment and other objects and can act intelligently under certain conditions through an autonomous behavior. Some examples of Smart Objects in our daily life are Smartphones, Tablets, Smart TVs and even some cars.

According to [3,18], Smart Objects can be classified in three dimensions, each of one corresponds to a type of intelligence. This way, through the three dimensions of an object we can determine its intelligence, what type of Smart Object it is and comparing it with others.

The aforementioned dimensions are:

- **Intelligence level**. It describes the amount of intelligence of the object. It is composed of three categories: The **management of information**: capacity to process the information captured by sensors, readers or other techniques; The **notification of the problem**, which is the possibility to inform its owner when a certain problem occurs or an event takes place, like a descend on the temperature. Lastly, the **decision-making**, which is the capacity to make decisions without requiring external control.
- **Location of the intelligence**. This one is composed of two categories: The first one is the **intelligence through the network**. It consists on an object whose intelligence depends totally on an external agent, which can be a network to which the object is connected to. The other category is the **intelligence in the object**. Objects belonging to this category compute everything by themselves, that is to say, they manage of the intelligence involved in the process.
- The last dimension is the **addition of the intelligence level** and it is composed of two categories: **intelligence in the element** and **intelligence in the container**. The first one encompasses all the objects that can only manage information, notifications and/or decisions and if they contain other components, these cannot be considered individual objects. For instance, a sensor. On the other hand, the intelligence in the container is not only capable of managing information, notifications and/or decisions, it can also keep working as an element or object even if some of its parts are separated from it. An Arduino board with at least three sensors would be a part of this group. If one sensor is taken from it, it could still work as a container.

This way, we can classify Smart Objects in its three dimensions. For this research, we will connect sensors with a level of intelligence, management of the information and notification of the problem, with the location of the intelligence through the network and the two categories of addition of the intelligence level.

3.3. Model Driven Engineering

Model Driven Engineering or MDE [29], appeared to solve software development problems. These problems are the low quality of the developed software, the breach of the budget and planning and an increase on the maintenance cost. These problems were already present in the 1960s and are still present, as it can be seen in [30,31]. The solution to this issues can be obtained through the automation or semi-automation of processes, something in which MDE is quite popular [32]. With this we manage to reduce the complexity of the design and the implementation, which helps obtain a much more reliable software and with more sophisticated functionalities [33]. Through the use of MDE we can increase the abstraction over the third generation programming languages (C++, C#, Java...). This abstraction is achieved through the use of models. They offers the use of a concept much closer to the problem's domain by converting the elements of the domain into one or more models [34]. This model makes easier for us to create a Domain Specific Language (DSL). By using this DSL, we manage to increase the abstraction of the problem, which implies an increase of the productivity [33].

For this proposal we use the Model Driven Engineering to create a higher level of abstraction and allow the creation of a DSL that makes easier to generate applications to interconnect heterogeneous object in a quick and simple way.

3.4. Domain Specific Languages

A Domain Specific Language or DSL is a language, commonly declarative, that makes calls for sub-processes in order to solve a certain problem within a specific domain. The DSL have a great power of expression [22,34]. The

main advantages that the use of DSL can offer are the increase of the productivity, the lesser chances of errors, their easy maintenance [35], their portability, the knowledge of the problem's domain and their reutilization for different purposes [22,36]. On the other hands, they present handicaps like a worse efficiency than native codification and a higher difficulty to create the domain and construct the DSL [22,35]. However, the Domain Specific Language are very important in Model Driven Engineering [34].

For this proposal we offer a Domain Specific Language that allows us to create applications to interconnect heterogeneous objects through an abstraction over the native codification of these. This, along with the graphic editor, will grant inexperienced users without knowledge on the domain of heterogeneous objects the possibility of creating the aforementioned application.

3.5. Internet of Things platforms

Nowadays, there are several platforms to interconnect devices. Among them there are some oriented towards business world, like the service providers: Xively [37], Exosite [38], SensorCloud [39] and Etherios [40]. Others are aimed at investigations, like Paraimpu [41], QuadraSpace [42] and SIoT [43]. We can find many publications about Paraimpu in many places. These detail the main problems of IoT. Lastly, we have the networks that are open to the users but are still in beta state, which can be only accessed with invitations or previously approved accounts. Some examples of this are ThingSpeak [44], Sensorpedia [45,46], SenseWeb [47,48], Evrythng [49] and Open.Sen.se [50]. Other example is Nimbits [51], an OpenSource platform with Apache License Version 2. In this section we will describe Paraimpu, Xively, ThingSpeak, Nimbits, SIoT and Open.Sen.se. The choice is clear: these are the ones that offer more data for research purposes due to their functionality. The other networks were not accessible due to their beta state or of the restricted information they provide. The six networks we have studied share something in common: they can generate triggers to interconnect two objects based upon a condition. Simple and limited triggers: one condition, one action.

3.5.1. Paraimpu

Paraimpu is a platform that can integrate several Smart Objects in one network that is capable of managing heterogeneous data, sharing it and connecting different sensors. This platform allows users to connect, create and share applications to connect objects. It is compatible with data in XML and JSON formats, but also with chains of characters and numerical data through the HTTP protocol. It separates objects in two types: sensors and actuators. The first ones are objects than can capture data and send it to the platform, while the seconds are objects that can perform an action. They provide the users with predefined classes to use 10 sensors and 13 actuators. Among these there are classes to use Arduino, Twitter, Pachube and Foursquare. It can establish a direct connection between two heterogeneous objects as long as they incorporate the platform's message sending mechanism. To generate the applications, the user will establish the condition he wants to be met by indicating the name of the sensor, its value, its condition and the result to be executed: "*Match: NameOfTheSensor.value > 25*" y *"Replace: 'today is <%(new Date().toDateString())%> and we have <% NameOfTheSensor.value%> <% NameofTheSensor.unit%>"*. It allows users to configure over the level of privacy when sharing the sensors [7,25,41,52].

Besides, as it can be seen in the examples they provide, the user must know certain rules of programming and use a quite programming-oriented language, such as the creation of an object (new), the insertion of a script (<%%>) or the access to a property or a method (.). They managed to drastically reduce the complexity of programming an application, but the user still needs to use methods that are almost like programming. It must be highlighted that they can only be used to create conditions, reducing the number of possibilities that other control structures can offer to control objects.

Midgar combines the use of a DSL and a graphical editor to obtain a higher level of abstraction in the application development process. A higher level of abstraction means that users of the platform need to have less technical knowledge and programming skills. Additionally, it offers more than just conditionals, it can also create loops, establish the time of the application's life or its launch, with the possibility of establishing several actions on each node or setting the device in a state of rest for any period of time.

3.5.2. Xively

Previously known as COSM and Pachube, Xively is a supplier of services in the cloud. This IoT network offers the possibility of uploading data from different Smart Objects or objects using the libraries they provide. To use them, one must use the API Key that is given to each new user. With it, one can limit the use of certain REST action in a device. They give you a total of ten libraries, among which we should highlight Arduino, Java and Android libraries. By using them, a user can upload data from different objects to his profile. He can also add triggers to this new device, so, once a certain condition is met, the device will send a HTTP request to a service and communicate the action to it by using a REST method [37].

Xively offers many privacy settings and access to REST through other service, and it also makes easier to connect objects through its trigger creator. These are focused in users without knowledge in programming. However, Xively only permits to create an action for a certain condition. This greatly reduces the chances of creating applications. On the contrary, Midgar allows the creation of actions and conditions without limitation. It allows you to execute various actions based on one or more conditions. As well, you can also set a condition based on another condition. Thus, it provides the possibility of interconnecting an infinite number of objects between them under different conditions that best suited to the operation of the application.

3.5.3. ThingSpeak

ThingSpeak is an IoT network for connecting objects. It offers tutorials on how to connect different objects and services. The first thing to do is creating a new device by

selecting its type (Arduino, Netduino, ioBridge...) and imputing its access data (IP, port, subnet mask...). It allows us to create a new channel. Channels are used to upload data from the connected devices, showing them in a graphic way and access their data through a REST request and downloading it in XML, JSON or CSV format. Once the data is uploaded, it creates interactive charts to visualize the data. It also offers a privacy system to establish whether the data is accessible to everyone or only to the user [44].

This platform provides a good system of data display and interactive charts. However, to connect different devices it is necessary to download the raw data and process it, which requires lots of work and programming knowledge, because you have to develop the application. Midgar solves this problem with its DSL and its graphic editor, making easier to interconnect data between devices in a quick and simple way, without requiring programming knowledge, just the item we want to connect.

3.5.4. Nimbits

Nimbits is an ecosystem that is based in users downloading the Nimbits Server (Google App Engine Version or Ubuntu Version) so that different users can connect to this server. This way it allows users to create their IoT servers. The Nimbits Server provides APIs Open Source to connect different objects as Arduino and Android. The server allows to use a REST web Service or the XMPP protocol. This offers the data in JSON and XML format. The data are stored as "Data Point". The "Data Point" contains numbers, text, timestamp, GPS coordinates, XML or JSON data and the alert state. Objects exist in Nimbits Server as "Points". The "Points" allow specifying filters, the type of the "point", the objective, the life of the "point", the unit of measure and the audience. The user can create "triggers". A "trigger" executes one action when a specific "point" receives new data. The "triggers" can be used to perform computation over the new and old values. Once the "point" is defined, the user can create subscriptions to it. Subscriptions can warn when a "point" receives new data, throw an alarm or a determined value is decrement and increment. Other option is the creation of alarms. Alarms can be set to warn according to some existing states. All this options can be set in the web editor.

Nimbits is a downloadable server. This is a good idea to create private IoT networks. Midgar shares this idea. Furthermore, Nimbits allows the XMPP protocol to be used in this platform with different message services. However, Nimbits limits the "triggers" to three parameters. In the other hand, Midgar offers unlimited parameters to create a trigger.

3.5.5. SIoT

SIoT is a platform for the Social Internet of Things [16,53]. SIoT is built over the ThingSpeak core. This platform is based on previous research in Social Internet of Things as [15,16,53]. It is a RESTful application and supports JSON, XML and CSV formats. To send data, devices can use GET or POST methods. This platform allows creating friend lists between objects because objects can create their own relationships.

This IoT network allows the creation of channels. In these channels, you can give permissions to relationship types or to indirect location. You can choose which data will be stored from your devices. The API offers sixteen fields to send data: temperature, current, voltage, power, mac address, latitude, longitude, humidity, pressure, alarm and four custom field for others data. A channel can be placed in a public area, at work or at home. When you select an existing location, the relative fields will be automatically filled.

To send and work with any channel, we must use the API Key. This is unique for each channel.

SIoT is a great social IoT platform. It is a social network for objects and these objects can create their own relationships. The platform offers three different formats to read the data (XML, JSON and CSV) and extra support for any data type. However, unlike Midgar, it does not provide a graphic editor to create the interconnection between objects, and it also operates across links. Thus, the objects must read the data and execute the actions. Midgar offers a solution by only sending the data that the objects request.

3.5.6. Open.Sen.se

Open.Sen.se is a beta platform for Internet of Things. The platform supports HTTP Posting, XMPP Posting and CoAP Posting protocols to send data and HTTP Polling, HTTP Push, XMPP Push and CoAP Push for receiving data.

You can add channels to send and receive data to and from this platform, such as physical devices (Arduino, trackers ...) or web forms (entering data manually, web services or others IoT platforms). In these channels, you create applications to process your data. The applications can be processed, triggered or displayed. With these applications you can process raw data that the application receives. You can perform actions when a specific event occurs or you can visualize the data. The platform supports any data type: numbers, text, binary or even incomprehensible content. All these data can be seen in the Senseboard. Users can view, control and share your data in your public Senseboard or as embed data in other sites.

Open.Sen.se offers a great platform for IoT and it supports different protocols but only permits to create interconnection between objects and web services. Furthermore, it creates different graphs in real time to show the history of the data. However, it only provides a basic support to interconnect objects. In our case, Midgar provides with a graphical editor that permits to create any idea to interconnection between objects, between web services or between objects and web services.

4. Case study: Midgar

Midgar is an Internet of Things platform specifically developed to investigate the problems of these platforms and the interconnection of objects. In this paper, we try to find a solution for the generation of applications that interconnect heterogeneous objects. In this section, we will describe the Midgar platform and the proposed solution for the mentioned problems.

4.1. Generation of applications for the interconnection of heterogeneous objects

With the presented Domain Specific Language (DSL), the user will be able to interconnect different objects in a simple way. Thanks to the graphic editor, the user can create the application without needing programming knowledge. To interconnect them, the user can use the graphic editor to write the application by using the DSL created for the case. The user will not be forced to use the given editor if he does not want to. However, as it will be explained in Fig. 1, this will help to optimize the creation of the DSL. Thanks to the DSL, a user can describe all the flow that he wants to be performed by the application. Therefore, it will be possible to connect many objects and check any of the parameters of their service and its data, as long as these services are registered in the platform. The user can read data from many objects and send certain actions to these objects or others, as many actions and conditions as the user wants.

Actions can differ from one object to another. These are registered at the same time as objects and services. An object can have an unlimited number of actions and services. Also, an application that interconnects objects can have an unlimited number of connected objects or actions to perform.

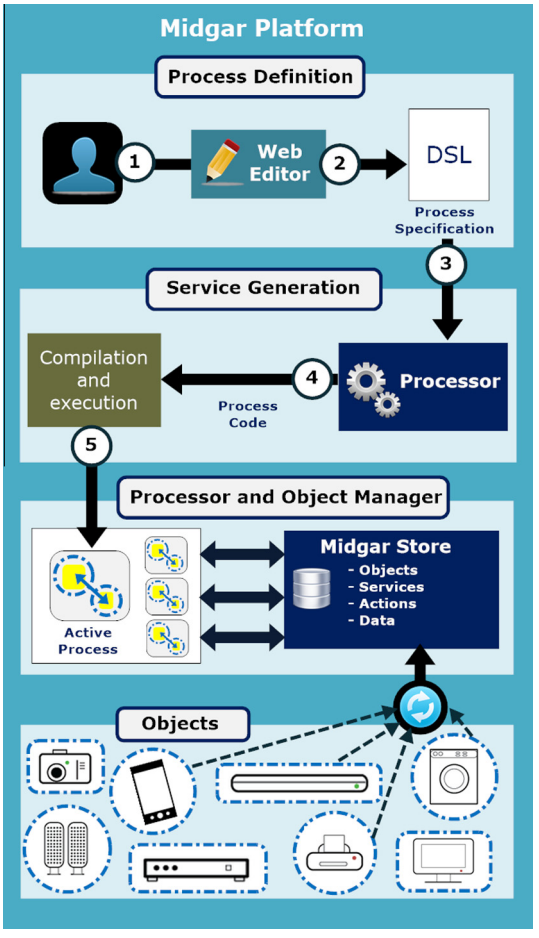

Fig. 1. Architecture of the Midgar platform.

4.2. Midgar's architecture

The system's architecture can be divided in four layers, as it can be seen in Fig. 1: **Process Definition, Service Generation, Processor and Object Manager** and **Objects**. Each one is a process within the global set of the infrastructure. So, the first layer, **Process Definition** (Fig. 1), encompasses the user's process. In this layer, **the user** (Fig. 1.1) must define the purpose of his application by using the **Web Editor** (Fig. 1.2). When the user concludes the definition of the application, the **Web Editor** generates the model of the application in the **DSL** (Fig. 1.2). This contains the information that the web editor generated in XML format. The information is moved to the second layer (Fig. 1.3): **Service Generation** (Fig. 1). This layer processes the information in the **Processor**. When the information is processed, the **Processor** creates a Java application with all the information (Fig. 1.4). This application has all the functionality that the user defined in the **Web Editor**. Then, **Midgar** compiles and executes this new application (Fig. 1.5). This last step is part of the third layer: **Server Layer** (Fig. 1). Afterwards, the application created by the user keeps working in the server (Fig. 1: Active Process), performing the task defined by the user. As long as it is functioning, it will directly communicate with the server and the data base (Fig. 1: Midgar Store). Moreover, if required, it establishes the messages that must be sent to the object the next time it is connected. This layer also contains the service that must be connected to the Smart Objects that interact with Midgar. These objects must implement a certain transmission of messages to keep the connection with the server. By processing these messages, the server obtains the data from each object. Smart Objects are found in the last layer: **Objects** (Fig. 1). They implement the message interface to keep a permanent, bidirectional connection with the server.

4.3. Implementation

In this sub-section we will describe in detail the different components and layers of the platform and their interaction. We will introduce the functionality and detail the Domain Specific Language we have created. All these components belong to the first layer and serve as a link to the second one. This last layer will be explained including the DSL processing and how it generates the application hosted in the server. We will keep talking about the Processor and Object Manager layer and its two components: The process server and the data storage. Finally, we will indicate how the connection of objects to the platform works.

4.3.1. Graphic web editor

Midgar platform includes a graphic editor (Fig. 2). The graphic editor can be used as an alternative to model applications. The editor is based on visual modeling; the user selects the available elements and connects them together

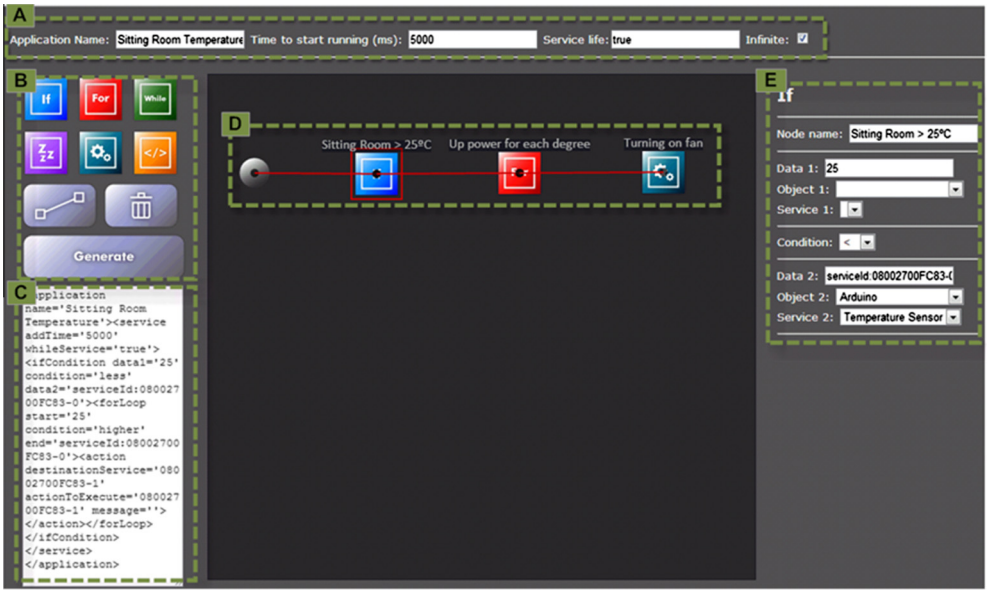

Fig. 2. Editor for the generation of object-interconnection applications of the Midgar platform.

to form a flow. The editor can be divided in five different areas.

The first area is shown in Fig. 2A. This area contains the data of the application we want to create: name time lapse from the moment of its creation to the moment in which it is executed, as well as the life time of the application. In this last one, we can set an infinite execution time by clicking on the checkbox.

The second area (Fig. 2B) contains the selection panel with the buttons that are used to add an element to the flow of the application. From left to right and from up to down, they are: The "If" button, which includes a condition for the flow: the "For" one, which allows the generation of loops that will be executed as long as a certain condition is met; "While", a loop that is very similar to "For" but gives support to the time-based conditions; "Sleep", which makes possible for the application to sleep during a certain amount of time and the "Action" one, the final and most important element, because its inclusion makes possible to send and order (along with a message) to an object to execute.

"Insertion of code", using this option allows the user to directly develop programming code in Java 6. This is inserted directly into the application. Java is used as base language for the generation of applications. The "Insertion of code" component is very adaptable to future solution extensions. If we decide to change the programming language of the generated applications, e.g. change java language for Python or Ruby. The editor could continue to use the same "Insertion of code" component, but users would have to include Ruby or Python code snippets. This component allows users with programming skills to extend the functionality of their applications.

Thus, it seeks to provide the possibility of customization to users with software development skills. The next button is the "interconnection" of nodes. It creates the connection (red arrow) between different nodes. It creates the flow that we want for the application. In the example, it can be seen how it connects to the beginning to the "if" node and this to the "for" node to finish in the "action" node. The interconnection allows the user to define this flow. Next to it there is the "Erase" button, which allows us to eliminate any element of the flow. Finally, the "Generate" button is intended to generate the application and visualize the DSL in the editor.

Fig. 2C contains the visualization of the DSL corresponding to an application created by the user with the editor. The working area is located in the center of the interface. Here is where the application is created and the flow connection is performed. The application flow always starts from the gray ball, as shown in Fig. 2D. The application shown in the example checks the value of the temperature of an Arduino micro-controller. If its temperature is over 25 °C, it creates a loop that raises the speed of the engine connected to the Arduino, which controls a fan located in the same room.

Finally, Fig. 2E shows the questionnaire area. Here we can visualize the questionnaire belonging to the selected node in the work zone. In this case, the select node is an "If" one.

4.3.2. Midgar's Domain Specific Language

The Domain Specific Language created for the definition of applications that interconnect devices has a total of six nodes. Four of them correspond to programming language structures, one corresponds to the insertion of direct code and language and other that corresponds to the action that has to be executed. The first four nodes are the conditionals, two loops and the sleep node. With these four, the user can create the flow of execution of any program, insert conditionals for it to act only under certain conditions, include a loop for executing tasks repeatedly or control the

flow of the program execution time by adding pauses or waits. To create the program flow through the use of the DSL, the user must nest one inside another if he wants them to be executed when the parent condition is met. If the user wants to execute one after another, he must simply place them in a row.

The DSL must always have a predefined header containing the name of the application and the waiting time (in ms) for the application to be executed, starting from the moment in which it is created. The other parameter indicates if the application must be executed endlessly of just during a certain amount of time.

```
<application name='MidgarTest'>
    <service initialTime='5000' lifeService='true'>
        ...
  </service>
</application>
```

The following code corresponds to one of the first developed mobile applications. The data obtained from the first service corresponded to the *Y* axis of the mobile phone accelerometer. When this was almost vertical and surpassed the aforementioned value, a message was sent to another mobile phone. In this first example, an action was executed whenever a condition was met. If the result returned by the second service of the *e6644a22aae2cec6* device is bigger than eight, the zero service of the *c3b9f28c24f2be8b* device executes the second action (actionToExecute) and sends a *"This action is a popup"* message. In this case, it executes the second action of the device, which is a message that shows the received message.

```
<ifCondition data1='serviceId:e6644a22aae2cec6–2'
  condition='higher' data2='8'>
      <action
  destinationService='c3b9f28c24f2be8b-0'
  actionToExecute='2' message='This action is a
  popup'> </action>
      </ifCondition>
```

Another example would be using a loop to send several actions in a row, but with a pause of a few seconds between each one. The following example corresponds to a developed application that checks the temperature of a sensor placed in an Arduino micro-controller. When this one surpassed the 25 °C, it sends a vibration (2500 ms per each extra grade) to the user mobile phone. This service could be useful to control places in which the temperature constantly raises and inform the controller in a not very intrusive way. Another use for this could be in the kitchen. We could send the warning to another Arduino micro-controller instead of a mobile phone, so an audible alarm could alert the cook. As can be seen in the code, when the value of the second service of the *08002700FC83* object surpasses 25, it executes a loop that goes from 25 to the number returned by the same service. Once it enters, it executes the action from zero with a message of 2500. After executing this action, it executes another similar to the one in the previous example and then waits five seconds to send the same action to the device once more.

```
<ifCondition data1='serviceId: 08002700FC83–2'
  condition='higher' data2='25'>
    <forLoop start='25' condition='less'
  end='serviceId:08002700FC83-2' >
      <action
  destinationService='c3b9f28c24f2be8b-0'
  actionToExecute ='0' message='2500'></action>
      <action
  destinationService='c3b9f28c24f2be8b-0'
  actionToExecute ='2' message='Temperature
  >25°C'></action>
      <sleep start='0' condition='higher'
  end='5000'>
      </sleep>
    </forLoop>
</ifCondition>
```

As seen in the previous example, the two actions and the sleep one are executed within the body of the loop, which is only executed when a certain condition is met. That is why they are nested. However, the actions and the sleep are executed one after another, because they are not nested.

The user can also perform a "While" loop or even insert Java code directly in the application. In the second case, this tag becomes useful for a user with knowledge in software development that desires to add more functionality to the application. As it can be seen in the example below, the user can even define the parameters that he will receive from the services he chooses and his name. After this, he will insert a Java code of his choice. In the example, if the first value is bigger than the second, it will perform an insertion in the data base to send a message to the mobile phone. This will execute the action 0, which starts a vibration and sends a parameter of 4000, equivalent to 4 s in the case of the vibration.

```
<java params='serviceId:c3b9f28c24f2be8b-1,
  serviceId: 08002700FC83-0'
  paramsNames='temperatureMobile,
  temperatureArduino' javaSource='if
  (temperatureMobile >temperatureArduino){new
  BBDD ().insertAnswer ("c3b9f28c24f2be8b-1", 0,
  "4000");}'></java>
```

4.3.3. Service Generation

As Fig. 3 shows, this second layer is composed of two subprocesses. This layer receives the DSL created with the web editor in XML format. Here, the generator of applications receives the DSL that contains the information of the application we want to develop. The DSL contains all the information about the flow that must be followed by the service the user wants to create. It processes all the information nodes it contains, creating a tree with that information. Once the tree is generated, it goes across it,

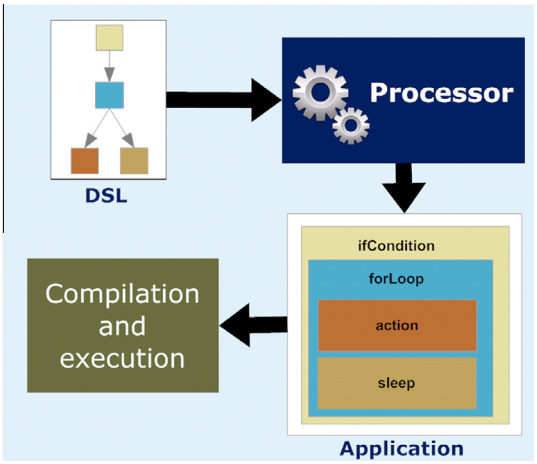

Fig. 3. Internal working of the Service Generation.

generating the application with the information contained in each node. The nodes can be conditions, instructions or actions, each one with its own configuration to define what it does. This way, two nodes of the same type can different tasks while being located in different parts of the code. After going across the whole tree and generating the application (a Java application, in our case), it is compiled and executed in the server.

4.3.4. Processor and Object Manager

This layer is composed of two modules that work together: the **process server** and the **data storage.**

Process server: In this layer we can find the user-developed applications. These are being executed constantly during their life cycle, making information requests to the server. They are continuously performing enquiries to the data base, and when the conditions established by the user are met, they insert the message that the server must send to the Smart Object and add a notification to alert the server about the existence of responses for that object.

Data storage: It centralizes all the requests, registers all the Smart Objects along with their services, stores all the data and actions of the services and sends the information to the applications created by the users. To do this, it uses a service-oriented architecture, as suggested in other research [11,21,24,25], as well as a REST architecture, as [26] suggests, for its many benefits and its easy use. Consequently, it becomes accessible to any object that implements the same message language than the Midgar platform, and by using REST, objects receive and send all the required messages. When it receives a message from an object, it processes it and stores the data in the data base. Then it checks if there are pending messages for the Smart Object. If there are messages, it generates the response with all the pending messages for the Smart Object and replies the requests. If there are no messages, it notifies the correct reception of the information.

4.3.5. Objects

Objects or Smart Objects that we want to connect with Midgar must implement a message specification that is understandable by the server. This specification defines the interface needed by the object to present its actions and services. The first step for an object to do is registering itself in the platform. By doing so, actions and services are registered in the Midgar data storage in order to allow users to use the services of the registered objects in the specification of the object-coordination process.

Services that can be offered by an object depend on the data that it can provide. For instance: a Smartphone can provide a service through a sensor; a car might inform about the number of kilometers or the consumption of fuel; the switchboard of a Smart Home could send data about each room temperature and humidity or communicate whether the doors and windows are open or closed.

Actions are what objects can do: a mobile phone could vibrate during a certain period of time, send a SMS, make a phone call or create a notification; a Smart Home could open and close doors and windows or turn on or turn off the heater or the humidifier. Once it is registered, an object can start sending data, including data of every service in each sending. Afterwards, it waits for a response from the server that can be just a confirmation of the reception of the message or a message with the actions to perform.

For this proposal, we used many mobile phones as Smart Objects, each one with different versions of the Android operative system, as well as an Arduino micro-controller as an object.

4.3.6. Android

The native application developed for Android mobile phones offers support starting from the 2.1 version, which corresponds with API7. This shows a list of the sensors of the mobile phone we are using and the values they capture in real time. The project makes it possible to modify those sensors we want to send in a simple way, as well as writing the registration message of the mobile phone services and actions. After that, the application is deployed in the mobile device, we push the registration button and run the application to start sending data to the data storage. This data will be used by the process server to execute the actions defined by the flow of the programs developed by the users in the graphic editor.

4.3.7. Arduino

The Arduino micro-controller (Fig. 4) was connected to the PC via a USB port. To send and receive messages we used a Java application that acts as intermediary for the Arduino. We facilitate its use, because the user needs not to use the C libraries or program at all. Besides, it makes it easier to create applications with Arduino thanks to its methods (which are very similar to the ones in the Android application) and uses a higher level language. It can also grant intelligence to the Arduino to transform it into a Smart Object when it becomes necessary for the tests. To be able to use it, we first need to install the sensors in the micro-controller, configure them in C (by using the Arduino own IDE) and then load the program. After that, we execute the Java application and it is ready to be used.

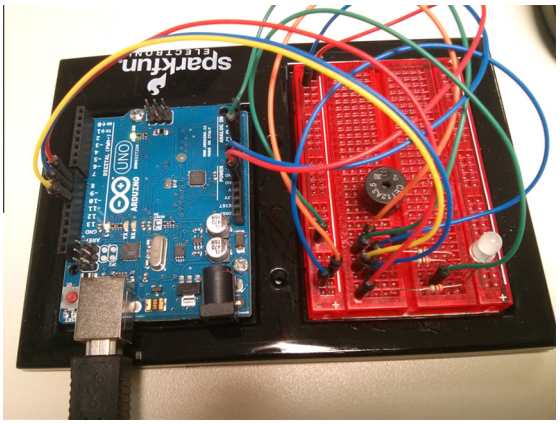

Fig. 4. Arduino picture during a testing.

First we register it and then the transmission of data starts automatically.

4.4. Used software and hardware

To develop this research work it is required the use of different types of software and hardware components:

- The Midgar server is based on the Ruby on Rails framework.
- The graphic editor was developed by using HTML5 and JavaScript. For this, the editor uses the HTML5 Canvas element. We used standard JavaScript to program the logic. We did not use any external library.
- The application generator module was developed using Java 6. Thanks to it we gained the multiplatform flexibility that could have been necessary in both cases

For the evaluation of the proposal we use a server and several components that served as a Smart Object:

- A server hosted in a dedicated computer with a Windows Server 2008R2 64 Bits operative system, a 3100 MHz Intel Core i3-2100 processor and 4 GB of RAM memory.
- Webrick web server.
- SQLite database.
- Four Smartphones: a Nexus 5 running Android 4.4.2, a Nexus 4 with Android 4.2.2, a Motorola with version 2.2.2, a Samsung Galaxy S with version 2.1 and a Samsung Galaxy Mini S5570 with version 2.3.6.
- One Arduino Uno microcontroller board based on the ATmega328. During the various tests we used: a temperature sensor, a speaker, a servomotor, a motor, several LEDs, two buttons and a potentiometer.

5. Evaluation and discussion

In this section we describe in detail the processes of evaluation selected and then we show the obtained results. In the first part of the evaluation we will describe the used methodology to perform the evaluations. After that, we will show and discuss the obtained results.

5.1. Methodology

The main objective of this evaluation process is to design and implement an evaluation process to validate that the solution proposed in this research is a step forward in the field of interconnection between Smart Objects. With the evaluation process, we intend to produce data that allows us to know if the proposed solution is useful for users of different profiles in order to model easily and quickly the interconnected application objects.

To analyse the relevant aspects of the proposal, an evaluation plan is specified. In this evaluation participated a range of different users. The evaluation aims to verify several aspects relevant:

- **Phase 1:** Check whether users of various profiles are able to use the proposed solution to complete an assigned task. This task corresponds to a basic use case to interconnect several objects.
- **Phase 2:** The users who have previously used the proposed platform will take a survey using the Likert scale [54] where they show their opinion about declarations. These declarations are based on the test, the project and the scope and possibilities that this research might have in the context of Internet of Things, more specifically, on the future of interconnection between objects.

5.1.1. Phase 1

First we defined a task to emulate a real scenario that requires three objects to be connected in order to achieve the goal.

The assigned task: The task consists in checking the temperature of the object A (Arduino micro-controller). When it surpasses the 25 °C, this should send a vibration of 2500 ms to the object B (Smartphone Nexus 4) and a notification to the object C (Smartphone Motorola Defy).

To be able to evaluate the simplicity of creation applications that interconnect heterogeneous objects, we decided to do a survey about user experience. This method is used in the software engineering field to provide information that effectively supports the decision-making [55]. This matches our case due to the impossibility of measuring the efficiency of the generation of applications through the graphic editor, as well as its potential. Because of this it is expected that, through these surveys, we can know whether the work we performed was successful or not.

For user profiles we chose software developers (SDev) and people interested in IoT (IoTU):

- SDEV members who are software developers and are not interested in IoT and do not work or worked with this technology.
- Members of IoTU, whether they are developers and not, but interested in IoT. They must work or have worked with this technology or at least they have this as a hobby.

Moreover, we did not take into account age or knowledge in the subjects. This is done to split between people who are interested in IoT and learn new knowledge about Smart Objects IoT. They can give the best opinion. Meanwhile, SDev group is composed by software developers who are not interested in IoT to evaluate the tools from a technical standpoint. Other reason for choosing these as profiles was the second contribution: attain an abstraction layer over the DSL that makes easier for anyone to create applications, it does not matter whether they are software developers or not. To perform the population sampling we used the snow ball method [56]. To do this, we started with people we knew, and then they started to invite more people for the test. We reached a total of 21 participants: 12 were software developers (SDev), while 9 were people interested in IoT (IoTU).

This way, we intended to evaluate the editor and the generator of applications from the point of view of both the developer and the user. This is important, because each one appreciates and demands different things, even though they are performing the same task.

The 21 selected users performed the test individually. (1–12) SDev (13–21) IoTU. At all times, they had a document with the objectives to be met. First, we explained the task to be undertaken by to the user and then we showed them the operation of the graphic editor. They were briefly shown the graphic editor: placement of the buttons, meaning, distribution, use, etc.

After that, the user began to model the application. Once the task was correctly completed, we stopped the timer to calculate the time spent by the user. It established a maximum time of 110 s $*$ 4 = 440 s (This is four times greater than the time spent by the creator of the supplied tool).

5.1.2. Phase 2

Once the user had finished the task of modeling the tool, moved to another computer and started to fill the survey, anonymously and without help (Table 1).

As a measurement method for the survey, we used the Likert Scale. We chose this one because it is the most used in the design of scales. We chose the 5-points Likert Scale giving the following options: 1 as strongly disagree, 2 as disagree, 3 as somewhat agree, 4 as agree, and 5 as strongly agree.

For the sample, we chose a size that represented the population and could offer significant results for the study [57].

To create the questionnaire we searched for ten declarations to ask for the user opinion on the use of the tool, its possibilities and its possible impact in Internet of Things and Smart Objects.

We did it to evaluate from the point of view of both profiles. However, we intended to offer the same tool and the same functionality to both of them. As a result, we evaluated them together.

The survey is made up of a set of 10 declarations that are shown in Table 1.

5.2. Results

5.2.1. Phase 1

Fig. 5 shows the time it took to each participant to complete the tasks. Finally, we show the average of all of them. Analyzing this table we can see that the fastest participant needed 93 s to finish. The biggest amount of time that a participant needed to make the application was 264 s. The average time between the 21 participants was 162.19 s. The standard deviation was 53.30.

5.2.2. Phase 2

In Table 2 we can see the responses sent by each user in an anonymous way. The table contains the responses from both the software developers (SDev) and the users interested in IoT (IoTU).

In Table 3 we present the descriptive statistics of all the set. Here, we can see the breakdown of each question: the minimum, the first quartile, the median, the third quartile, the maximum, the range (maximum–minimum), the range between quartiles and mode. Fig. 6 shows all this data in a Box and Whiskers Plot diagram.

By analyzing Table 3 and Fig. 6, we can suggest the following interpretations:

- Q3 has the highest minimum, in this case, 4 out of 5. This means that all the participants agree the declaration, at the very least.
- Q3, Q4 and Q6 are the declarations with highest median. From this we can deduct that most of the participants agree these declarations.

Table 1
Survey given to the users.

Question	Description
Q1	The user understands the functionality of the editor elements and their role in application creation process.
Q2	This editor allows to interconnect devices in simply, using a few clicks and without having to program code.
Q3	The editor approach makes it difficult to make mistakes while the user is modeling the applications.
Q4	This solution offers a fast way to developing the indicated task.
Q5	This solution provides assistance to create applications to interconnect objects.
Q6	The editor does not require the user to use complex programming skills, as in traditional application development.
Q7	The editor includes enough elements and functionality for the user to create a wide range of applications to interconnect objects.
Q8	This proposal is a positive contribution to encourage the development of services and applications that provide interconnection between objects.
Q9	Internet of Things and Smart Objects will benefit will be benefited by this solution.
Q10	This editor could be used to simplify the classic development process of software applications in other areas.

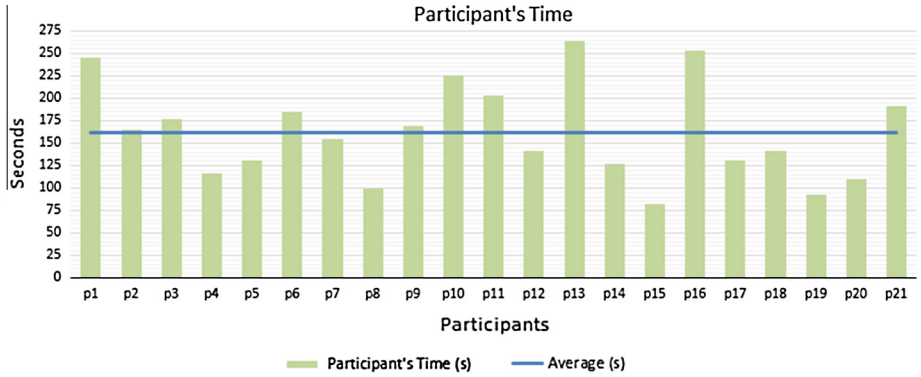

Fig. 5. Time to complete the task by the participants.

Table 2
Responses of the users for each question.

Id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
p01	3	4	4	4	4	4	3	3	3	4
p02	5	5	5	5	5	5	5	5	5	4
p03	4	5	4	5	4	5	3	3	4	3
p04	3	5	5	4	4	4	3	4	5	4
p05	2	3	5	4	4	4	3	4	4	4
p06	4	4	5	4	3	3	3	5	4	3
p07	3	4	5	5	4	5	4	4	5	5
p08	4	4	5	5	5	5	3	5	4	5
p09	4	5	5	5	5	5	5	3	5	4
p10	4	5	4	5	5	5	4	5	5	5
p11	5	4	5	4	4	4	3	5	4	5
p12	3	3	4	3	4	4	2	3	3	3
p13	4	4	5	4	5	5	4	4	5	5
p14	4	4	5	5	4	4	4	4	4	4
p15	5	4	5	5	5	5	5	5	4	5
p16	3	3	4	4	4	5	3	5	4	5
p17	5	5	5	4	5	5	5	4	4	5
p18	4	4	5	5	4	5	2	2	3	4
p19	4	5	5	5	5	5	5	4	5	5
p20	5	5	5	5	5	4	5	5	5	4
p21	5	5	4	5	5	5	5	5	5	4

- Q3 is the only question with a range of 1. This means that all the participants have the same opinion on this declaration. On the other hand, Q1, Q7 and Q8 have the highest range. This indicates that there is a big difference between the answers of each participant.
- According to the mode, we can observe that Q7 has a mode of 3. This shows that the answer chosen by most of the participants is *Neutral*. On the other hand, Q3, Q4, Q6, Q8 and Q9 have a mode of 5, which indicates that the most chosen answer for these declarations was *Strongly Agree*.
- Regarding Q1, Q7 and Q8, they have a range of 3, a minimum of 2, a median of 4 and a maximum of 5. Despite having some low values, Q8 is considerably well-rated because it has a quartile 1 of 4, an Inter Quartile Range of 1 and a mode of 5.

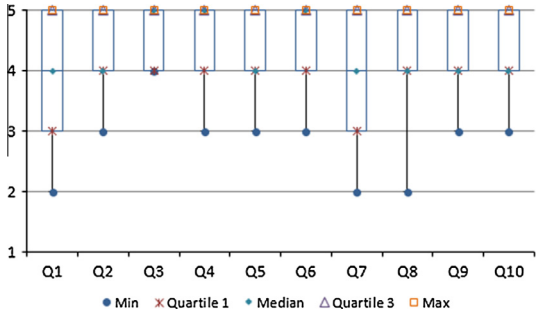

Fig. 6. Box and whiskers plot for each question.

In Table 4 we can see the frequencies for the answers to each question. Here we have a breakdown of each question: the number of votes for each decision and the percentage corresponding to both of them. Fig. 7 shows a bar graphic with the frequency of the answers in the set formed by all the profiles.

From Table 4 and Fig. 7 we can figure things that could not be figured before. These are the interpretations:

Table 3
Table with the general descriptive statistics.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Min	2	3	4	3	3	3	2	2	3	3
Quartile 1	3	4	4	4	4	4	3	4	4	4
Median	4	4	5	5	4	5	4	4	4	4
Quartile 3	5	5	5	5	5	5	5	5	5	5
Max	5	5	5	5	5	5	5	5	5	5
Range	3	2	1	2	2	2	3	3	2	2
Inter Qrt.-Range	2	1	1	1	1	1	2	1	1	1
Mode	4	4	5	5	4	5	3	5	5	4

Table 4
Frequency table for the general responses.

Question		Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Q1	#	0	1	5	9	6
		0	5	24	43	29
Q2	#	0	0	3	9	9
		0	0	14	43	43
Q3	#	0	0	0	6	15
		0	0	0	29	71
Q4	#	0	0	1	8	12
		0	0	5	38	57
Q5	#	0	0	1	10	10
		0	0	4	48	48
Q6	#	0	0	1	7	13
		0	0	5	33	62
Q7	#	0	2	8	4	7
		0	10	38	19	33
Q8	#	0	1	4	7	9
		0	5	19	33	43
Q9	#	0	0	3	9	9
		0	0	14	43	43
Q10	#	0	0	3	9	9
		0	0	14	43	43

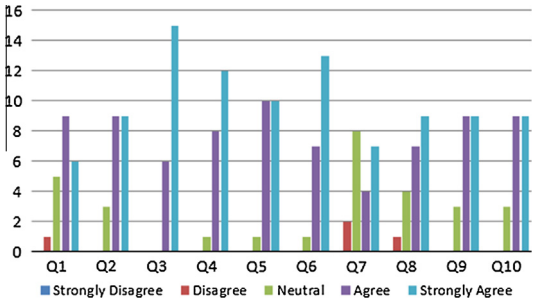

Fig. 7. Overall response distribution.

- Q5 has a 48% of the votes for *Agree* and *Strongly Agree*, while only the 4% voted *Neutral*. In numbers this means 10, 10 and 1 vote, respectively, which indicates that the majority of the participants agree this declaration.
- Q2, Q9 and Q10 have a 43% of the votes for *Agree* and *Strongly Agree* and a 14% for *Neutral*. These numbers correspond to 9, 9 and 3, respectively. This indicates that the majority agrees but there are an important percentage of people that are indecisive or do not believe in these declarations.

6. Conclusions

In this paper we present the Midgar platform, a novel proposal that provides a solution to handle heterogeneous Smart Objects and allows non-technical users to model and generate software applications that interconnect these Smart Objects. Traditionally, software application development that aims to connect heterogeneous Smart Objects is a relatively complex process, since in most cases requires solid programming skills and knowledge on different technologies.

Midgar develops a centralized solution at various levels, Smart Objects registration, publication of the catalog of actions associated with each Smart Object and the modeling and generation of specific software applications especially designed for interconnecting Smart Objects. One of the key aspects of the solution is that it increases the level of abstraction in the application development process using a graphics editor and own Domain Specific Language, which was designed for that specific purpose. Using the editor we managed to abstract the problem of having to develop an application in a programming language without needing programming knowledge.

To validate the platform technical effectiveness, it was used to interconnect multiple Smart Objects and make them work together. To validate that people could effectively use the proposed solution we perform various evaluations. The evaluations found that users were able to develop applications that interconnect objects correctly in an average time of **162, 19** s.

The evaluation process also included a survey, useful for users to evaluate various subjective aspects of the proposal. These aspects are related to the user experience while using the proposed solution, user answers are useful to know if the proposal allows modeling interconnections between objects in an understandable and agile way. The 90% of the declarations we posed to users obtained more than 75% of positive or very positive assessments. Even the lowest rated declaration obtained a 52.38% of positive or very positive assessments.

Noteworthy that most of the non-positive user evaluations were motivated by the limited development characteristics that the platform was using to model applications, the included features to model a large number of applications, but they are very far to allow to model any application imaginable. In future proposal extensions we will give emphasis on new modeling elements to extend the possibilities for application development, adding new mechanisms to interconnect nodes, and more complex and advanced tasks. In any case, these extensions

must respect the nature of the platform and avoid adding undue complexity to the modeled process.

In function of the results of the evaluations, we can say that the Midgar platform can be very useful to reduce the complexity of interconnected Smart Objects. This also allows non-technical users to be able to interconnect Smart Objects.

The communication and collaboration between Smart Objects order to achieve goals is one of the key aspects of Internet of Things. In the future it is expected that people will live surrounded by multiple Smart Objects. People increasingly have shown a greater ability to handle electronic devices. It is very possible that in the future the people want or need to adapt their Smart Objects behavior to their needs, defining collaborations between objects in their homes, workplace, etc. Midgar platform is a step towards this scenario where people both technical and non-technical set the collaboration between Smart Objects in their environment.

7. Future work

Internet of Things is part of the future and there is still plenty to investigate. As mentioned before, there are still many problems to solve. Below we list the ones that could be solved thanks to this investigation:

- **Domain Specific Language and graphic editor to generate Smart Objects**: Create a DSL and a graphic editor, similar to the one shown in this research, which generates the application to deploy in the desired object. This would make easier to create Smart Objects so they can be interconnected.
- **Insertion of boxes in the graphic editor with support for data analysis and fuzzy logic:** In this way, a user could generate a data analysis for objects and define their own boxes with fuzzy logic to realize checks.
- **Security and privacy in IoT**: Study of the security and privacy methods that any IoT network should have to prevent the violation of data transferred from and to the objects.
- **Scalability of IoT platforms:** Studies and tests different implementations of the IoT platform to check which one offers a bigger scalability.

References

[1] H. Gu, D. Wang, A Content-aware fridge based on RFID in smart home for home-healthcare, in: 2009, pp. 987–990.
[2] C. Hao, X. Lei, Z. Yan, The application and implementation research of Smart City in China, in: 2012, pp. 288–292.
[3] K.A. Hribernik, Z. Ghrairi, C. Hans, K. Thoben, Co-creating the internet of things – first experiences in the participatory design of intelligent products with Arduino, in: 2011, pp. 1–9.
[4] C. Han, J.M. Jornet, E. Fadel, I.F. Akyildiz, A cross-layer communication module for the internet of things, Comput. Netw. 57 (2013) 622–633.
[5] F. Telefónica, La Sociedad de la Información en España 2012, 2013.
[6] V. Georgitzikis, O. Akribopoulos, I. Chatzigiannakis, Controlling physical objects via the internet using the arduino platform over 802.15.4, Networks 10 (2012) 1686–1689.
[7] A. Piras, D. Carboni, A. Pintus, D.M.T. Features, A platform to collect, manage and share heterogeneous sensor data, in: Networked Sens. Syst., 2012, pp. 1–2.
[8] L. Tan, Future internet: the internet of things, in: 2010 3rd Int. Conf. Adv. Comput. Theory Eng., 2010, pp. V-5376–V5-380.
[9] G. Kortuem, F. Kawsar, D. Fitton, V. Sundramoorthy, Smart objects as building blocks for the internet of things, IEEE Internet Comput. 14 (2010) 44–51.
[10] I.W.S.Y.S.E.C. Akyildiz, A survey on sensor networks, IEEE Commun. Mag. (2002) 102–114.
[11] L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (2010) 2787–2805.
[12] N. Bulusu, D. Estrin, L. Girod, Scalable coordination for wireless sensor networks: self-configuring localization systems, in: Proc. 6th Int. Symp. Commun. Theory Appl. A '01), Ambleside, UK, 2001, pp. 1–6.
[13] G.M. Lee, J.Y. Kim, Ubiquitous networking application: energy saving using smart objects in a home, in: 2012 Int. Conf. ICT Converg., 2012, pp. 299–300.
[14] R. Roman, J. Zhou, J. Lopez, On the features and challenges of security and privacy in distributed internet of things, Comput. Netw. 57 (2013) 2266–2279.
[15] L. Atzori, A. Iera, G. Morabito, M. Nitti, The social internet of things (SIoT) – when social networks meet the internet of things: concept, architecture and network characterization, Comput. Netw. 56 (2012) 3594–3608.
[16] R. Girau, M. Nitti, L. Atzori, Implementation of an experimental platform for the social internet of things, in: 2013 Seventh Int. Conf. Innov. Mob. Internet Serv. Ubiquitous Comput., IEEE, Taichung, 2013, pp. 500–505.
[17] The US National Intelligence Council, Six technologies with potential impacts on US interests out to 2025, 2008.
[18] G.G. Meyer, K. Främling, J. Holmström, Intelligent products: a survey, Comput. Ind. 60 (2009) 137–148.
[19] M. Rothensee, A high-fidelity simulation of the smart fridge enabling product-based services, in: 3rd IET Int. Conf. Intell. Environ. (IE 07), IEE, 2007, pp. 529–532.
[20] A. Broring, P. Maue, C. Malewski, K. Janowicz, Semantic mediation on the sensor web, in: 2012, pp. 2910–2913.
[21] K. Gama, L. Touseau, D. Donsez, Combining heterogeneous service technologies for building an internet of things middleware, Comput. Commun. 35 (2012) 405–417.
[22] A. Van Deursen, P. Klint, J. Visser, Domain-specific languages: an annotated bibliography, ACM Sigplan Not. (2000).
[23] D. Guinard, V. Trifa, Towards the web of things: web mashups for embedded devices, in: 2nd Work. Mashups, Enterp. Mashups Light. Compos. Web, Madrid, 2009.
[24] A. Pintus, D. Carboni, A. Piras, A. Giordano, I. Elettrica, Building the web of things with WS-BPEL and visual tags web of things using service-oriented architecture standards, in: Fourth Int. Conf. Mob. Ubiquitous Comput. Syst. Serv. Technol. (UBICOMM 2010), Florencia, Italy, 2010, pp. 357–360.
[25] A. Pintus, D. Carboni, A. Piras, The anatomy of a large scale social web for internet enabled objects, in: Proc. Second Int. Work. Web Things – WoT'11, 2011, p. 1.
[26] D. Guinard, V. Trifa, T. Pham, O. Liechti, Towards physical mashups in the web of things, in: 2009 Sixth Int. Conf. Networked Sens. Syst., IEEE, 2009, pp. 1–4.
[27] D. McFarlane, S. Sarma, J.L. Chirn, C. Wong, K. Ashton, Auto ID systems and intelligent manufacturing control, Eng. Appl. Artif. Intell. 16 (2003) 365–376.
[28] C.D.Mf.A.A.V.A. Wong, The intelligent product driven supply chain, in: Syst. Man . . ., 2002.
[29] S. Kent, Model driven engineering, Comput. Comput. Soc. 2335 (2002) 286–298.
[30] E. Dijkstra, The humble programmer, Commun. ACM. 15 (1972) 859–866.
[31] E. González, H. Fernández, V. Díaz, General purpose MDE tools, IJIMAI 1 (2008).
[32] V. García-Díaz, H. Fernández-Fernández, E. Palacios-González, B.C.P. G-Bustelo, O. Sanjuán-Martínez, J.M.C. Lovelle, TALISMAN MDE: mixing MDE principles, J. Syst. Softw. 83 (2010) 1179–1191.
[33] B. Selic, MDA manifestations, Eur. J. Inform. Prof. IX (2008) 11–16.
[34] E.R. Núñez-Valdez, O. Sanjuan-Martinez, C.P.G. Bustelo, J.M.C. Lovelle, G. Infante-Hernandez, Gade4all: developing multi-platform videogames based on domain specific languages and model driven engineering, Int. J. Interact. Multimed. Artif. Intell. 2 (2013) 33–42.
[35] A.B. Reed, V. Halpern, J.E. Starr, Little languages: little maintenance? 1996.

[36] A. Van Deursen, Domain-specific languages versus object-oriented frameworks: a financial engineering case study, 1997.
[37] LogMeIn, Xively, 2013. <Https://xively.com/>.
[38] Exosite, Exosite, 2013. <Http://exosite.com/>.
[39] LORD MicroStrain, Sensor Cloud, n.d.. <Http://www.sensorcloud.com/>.
[40] Etherios, Etherios, 2008. <Http://www.etherios.com/>.
[41] A. Pintus, D. Carboni, A. Piras, Paraimpu, 2012. <Http://paraimpu.crs4.it/>.
[42] QuadraSpace, 2010. <Http://www.quadraspace.org/>.
[43] Department of Electrical and Electronic Engineering (University of Cagliari), SIoT, 2012. <Http://platform.social-Iot.org/>.
[44] IoBridge, Thingspeak, 2013. <Http://www.thingspeak.com>.
[45] Oak Ridge National Laboratory, Sensorpedia, 2009. <Http://www.sensorpedia.com/>.
[46] B.L. Gorman, D.R. Resseguie, C. Tomkins-Tinch, Sensorpedia: information sharing across incompatible sensor systems, in: 2009 Int. Symp. Collab. Technol. Syst., 2009, pp. 448–454.
[47] Microsoft, SenseWeb, 2008. <Http://research.microsoft.com/en-Us/projects/senseweb/>.
[48] A. Kansal, S. Nath, J. Liu, W.I. Grosky, SenseWeb? An infrastructure for shared sensing, IEEE Multimed. 14 (2007) 8–13.
[49] EVRYTHNG, EVRYTHNG, 2012. <Https://www.evrythng.com/>.
[50] Sen.se, Open.Sen.se, 2011. <Http://open.sen.se/>.
[51] I. Nimbits, Nimbits, n.d.. <Http://www.nimbits.com/>.
[52] A. Pintus, D. Carboni, A. Piras, Paraimpu: a platform for a social web of things, in: Int. World Wide Web Conf. Com-Mittee, 2012, pp. 401–404.
[53] L. Atzori, A. Iera, G. Morabito, SIoT: giving a social structure to the internet of things, IEEE Commun. Lett. 15 (2011) 1193–1195.
[54] R. Likert, A technique for the measurement of attitudes, Arch. Psychol. 22 (1932) 1–55.
[55] M. Kasunic, Designing an effective survey, 2005.
[56] R.M. Groves, F.J. Fowler, M.P. Couper, J.M. Lepkowski, E. Singer, R. Tourangeau, Survey Methodology, Wiley-Interscience, 2004.
[57] B. Kitchenham, S.L. Pfleeger, Principles of survey research Part 2: Designing a survey, IGSOFT Softw. Eng. Notes 27 (2002) 18–20.

Cristian González García is a Technical Engineering in Computer Systems and M.S. in Web Engineering from School of Computer Engineering of Oviedo in 2011 and 2013 (University of Oviedo, Spain). Currently, he is Ph.D. candidate. His research interests are in the field of Internet of Things, Web Engineering, Mobile Devices and Modeling Software with DSL and MDE.

B. Cristina Pelayo G-Bustelo is a Lecturer in the Computer Science Department of the University of Oviedo. Ph.D. from the University of Oviedo in Computer Engineering. Her research interests include Object-Oriented technology, Web Engineering, eGovernment, Modeling Software with BPM, DSL and MDA.

Jordán Pascual Espada is a Research scientist at Computer Science Department of the University of Oviedo. Ph.D. from the University of Oviedo in Computer Engineering. His research interests include the Internet of Things, exploration of new applications and associated human computer interaction issues in ubiquitous computing and emerging technologies, particularly mobile and Web.

Guillermo Cueva Fernández received his B.S. in Computer Science and Engineering and M.S. in Web Engineering in the University of Oviedo in 2009 and 2012. He is now working to obtain his Ph.D. degree. His research interests are the automotive area, mobile devices and data mining.