# A User-Oriented Language for Specifying Interconnections between heterogeneous objects in the Internet of Things

Cristian González García*, Liping Zhao, and Vicente García-Díaz

**Abstract**—We propose a user-oriented language to enable end users to specify interconnections between heterogeneous objects in the Internet of Things (IoT). Based on the idea of the use case specification technique in software engineering, our language provides end users with a natural language like syntax to allow them to specify *when* or under *what* condition they want *which* objects to be connected. To support this language, we have also developed a transformation mechanism that automatically translates users' specification into the source code. We have evaluated this language through an experiment and a survey. The main contributions of this paper are: (1) a simple natural language that enables the end-users to specify which objects to connect and when, and (2) a transformation mechanism that automatically translates users' specifications into source code and dynamically attaches the code to relevant applications. Our work represents a first step in bringing the IoT closer to their users.

**Index Terms**—Application Platforms, Service Functions and Management, Service Middleware and Platform, User experience

## 1 Introduction

THE vision of the Internet of Things (IoT) is to reach out for everyday objects in the real world and connect them to the Internet, thus achieving *anytime* and *anyplace* connectivity for *anyone* and *anything* [1], [2]. According to Cisco's white paper [3], by 2020 there will be 50 billion physical objects connected to the Internet.

Smart objects, such as smartphones, smartwatches and tablets, play a key role in the IoT vision as they are programmed with intelligent information and communication software. Thus when connected to sensors, these objects are able to perceive their context and location; with their built-in networking capabilities, they can communicate with each other, access Internet services and interact with people [4]. 'Conventional objects', such as sewing machines, exercise bikes, electric toothbrushes, washing machines, electricity meters and photocopiers, can have a 'digital make-over', that is, by adding the capabilities of digital objects, to enhance their functionality [4]. With digitalisation (digital objects) and sensors, we can connect both smart and non-smart objects to the Internet [5], make them communicate with each other and create value-added, intelligent applications such as 'Smart Homes' and 'Smart Cities' – the dream of the IoT [6]. However, sometimes, the need of good or special practices because of the different necessities or limitations of the hardware, like the battery [7], [8] and the energy consumption [9], [10], the low computing power [11], the centralised control of many devices [12], or a standardisation, data management or security, among others [13], [14].

In the IoT, however, things, which can be smart and non-smart [5], are usually diverse, as they are made by different manufacturers, serve different purposes, contain different physical components, use different interface standards, have different communication protocols, and embed different software technologies, and so on [15]–[17]. These differences inevitably result in heterogeneous objects that cannot directly communicate [15], [18]. Although global standards on the IoT may ease the heterogeneity problem, creating such standards is currently a major challenge to the IoT [4], [19], [20].

A common solution to this problem combines the principle of information hiding and encapsulation with the concept of Service-Oriented Architecture (SOA). This approach provides each object with a service, which acts as a communicator (i.e., interface) for the object and hides the object details from the client [15]. By using such an approach, objects heterogeneity is hidden from the service consumers, allowing applications to use those objects via standard services.

Yet, the IoT-based applications are dynamic. We quote this example from [15]: 'A device such as a Bluetooth smartphone might become unavailable to a system as soon as it moves out of range. Regarding autonomy concerns, a simple sensor cannot perform its task anymore if its battery is depleted. As a consequence, a system hosting IoT-based pervasive applications must be highly dynamic to manage the devices, which continuously leave or enter the system.' This kind of application, therefore, cannot be fully described beforehand due to non-deterministic nature of service availability.


- *C. González García is with Department of Computer Science, University of Oviedo, Oviedo, Spain. E-mail: gonzalezcristian@uniovi.es.*
- *L. Zhao is with School of Computer Science, the University of Manchester, Oxford Road, Manchester, M13 9PL, United Kingdom, E-mail: liping.zhao@manchester.ac.uk.*
- *V. García-Díaz with Department of Computer Science, University of Oviedo, Oviedo, Spain. E-mail: garciavicente@uniovi.es.*

**** Corresponding author***

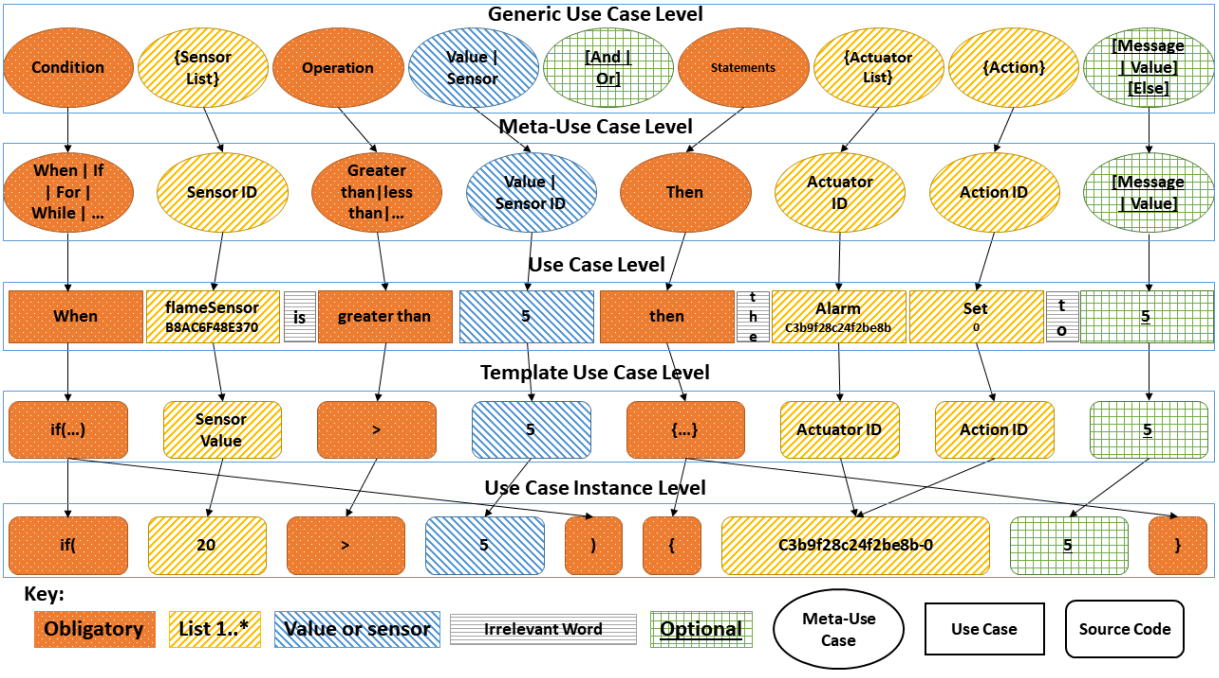



**Figure 1 The syntax and structure of MUCSL**

While dynamic SOA [21] offers a promising solution to this problem by allowing applications to react to service arrivals and departures according to their environment, developing dynamic applications in this fashion is a specialised job, assuming programming skills and software development knowledge [22]. This assumption seems to be at odds with the vision of the IoT, which inspires pervasive connectivity for anyone and anything at any time and space. Indeed, the IoT-based applications should be pervasive and made for the mass, rather than the few. To take the advantage of the IoT, businesses and individuals without professional knowledge should be able to define their own applications and decide when and which of their objects should be connected to provide their desired services.

With this motivation, we propose a user-oriented language to enable end users to specify interconnections between heterogeneous objects in the Internet of Things. Based on the idea of the use case driven approach in software engineering [23], our language provides end users with a natural language like syntax to allow them to specify *when* or under *what* condition they want *which* objects to be connected. To support this language, we have also developed a transformation mechanism that automatically translates users' specification into the source code.

The main contributions that can be identified from this paper are:

1. A Domain-Specific Language (DSL) very similar to natural language.
2. A DSL that enables the end-users to specify the interconnection among objects in an easy and simple way without programming.
3. A transformation mechanism that automatically translates users' specifications into source code that includes all the necessary logic.

Although there are already some related works, there is a great absence on DSLs in the IoT and more specifically in the scope of this same work [24] Thus, our work represents a first step in bringing the IoT closer to their users. Here, users can describe the interconnection that they need between the objects. This interconnection includes conditionals, loops, events (using the different structures), the selection of objects and the messages that you want to send to an object (actions). Thus, using the different structures, users can create the interconnection among objects with a small intelligence or based on different decisions.

We call our specific language **MUCSL** (Midgar Use Case Specification Language)**. Midgar** is an IoT platform developed in our early work [16]. Midgar is necessary because the objects have to be registered in this platform and have to use the message system described in the platform to create the interconnection, being Midgar our case study to demonstrate this research. We develop MUCSL, which is the contribution of this work, as a new layer on top of Midgar. The main difference with this previous research is that now, we have developed a text DSL, which is very close to natural language. This new DSL facilitates the conversion of use cases, according to some rules, to the automated source code generation. This being opposed to the previous research in which users had to learn a graphic DSL to develop the applications.

To continue, the next section details our language and its transformation mechanism. Section 3 presents an experiment on the utility of our approach and evaluates its usability by analysing the experimental results. Section 4 overviews other approaches that are related to our work, and finally, Section 5 concludes the paper by summarising its contributions and outlining future work.

| Category | | Words |
|---|---|---|
| Obligatory | Condition | If, for, when, while, meanwhile, as long as |
| | Comparison Operators | Greater than, more than, less than, lower than, minor than, equal, equal to, greater or equal to, equal or greater than, less or equal to, equal or less than, more or equal to, equal or more than, lower or equal to, equal or lower than, minor or equal to, equal or minor than, different from, set to, >, <, >=, <=, ==, != |
| Conjunction | | Then |
| Value | | Text, number, character, Boolean, sensor ID |
| ID | | Alphanumeric, - |
| Optional | Conjunction | And, or |
| | In another case | Else, otherwise |
| | Message | Text, number, character, Boolean |
| Others | Irrelevant Words | The, a, an, are, is, be, to, will, should, shall, can, could, would |
| | Punctuation | ‘,’, ‘,’, ‘.’, ‘:’, ‘;’, ‘’’, ‘’’, ‘””, ‘””, ‘«’, ‘»’ |

**Table 1 Keywords of MUCSL**

## 2 MUCSL: A User Oriented Language

### 2.1 Language Syntax and Structure

**MUCSL** has a simple lexicon consisting of ‘when’, ‘if’, ‘for’, ‘while’, ‘sensorID’, ‘then’, etc, defined at the **Meta-Use Case Layer**, as Figure 1 shows. MUCSL uses the syntax of conditional statements and allows the users to define their intended interconnection through simple conditional statements, such as ‘If condition then action’ and ‘While condition then action’.

The overall structure of MUCSL consists of five layers, as Figure 1 shows. The top layer, **Generic Use Case Layer**, provides the vocabulary for end users to define the sentence structure. This layer consists of six categories of words (see Table 1): obligatory, conjunction, IDs, values, optional, and other. These groups of words can be composed of simple and natural sentences that can be interpreted in a sequential way: First, we have the condition to indicate the beginning of a sentence. Second, we compare the value of a sensor or a list of sensors (to facilitate the use of numerous objects) with a numerical value to establish the true or false of the condition. Then, we have the option to add ‘AND’ or ‘OR’ in the condition if more sensors are going to be compared. This allows users to create and combine numerous sensors with different conditions in an easy way. The next word is the statement that we use to separate the condition of the clause. The statement contains the list of actuators with one or more actuators and the action that the user wants to do on those actuators. Afterwards, we have another optional word, indicating an alternative action either to ‘turn on’ or ‘turn off’ the actuator.

The second layer, **Meta-Use Case Layer**, provides possible words that some part of the structure needs. For example, the condition only accepts ‘When’ or ‘If’. The list of sensors needs one or more sensor IDs. The comparison operators can be ‘greater than’, ‘less than’, ‘equal to’, ‘greater or equal than’, ‘less or equal than’, ‘different from’, and so on. The value can be an exact value like a Boolean, a number, a text message, a character, a numeric value, or another sensor to obtain a value. For instance, maybe we have to turn on the light when the two or more photoresistor sensor of our living room has less value than five or maybe when the value of this is less than another photoresistor sensor that we have outside the house. The optional value ‘and’ or ‘or’ serves to us to concatenate more conditions. In this example, we did not use it. After, the statement must be the word ‘then’. This word indicates the beginning of the actions that the user wants to do when the conditions are accomplished. The next words are the list of one or more actuator IDs and the action ID that the user wants to activate in these actuators. Finally, we have the optional message or value that may be, the user must send to the action. After all, we can type the clause ‘else’ or ‘otherwise’ to do an action otherwise, when the normal clause will not accomplish.

The third layer, **Use Case Layer,** consists of user-defined use case instances. For example, in Figure 1, the statement ‘When the B8AC6F48E370 is greater than 5 then the C3b9f28c24f2be8b 0 to 5’ suggests that the object ID will be used instead of the name because we want to make sure the correct object is called to take the correct action. Users can see the unique IDs of their objects on the Midgar platform using the provided RESTful service. After mapping the IDs to the object names, the statement ‘When the B8AC6F48E370 is greater than 5 then the C3b9f28c24f2be8b 0 to 5’ will be translated into ‘When the flameSensor is greater than 5 then the alarm will be set to 5’.

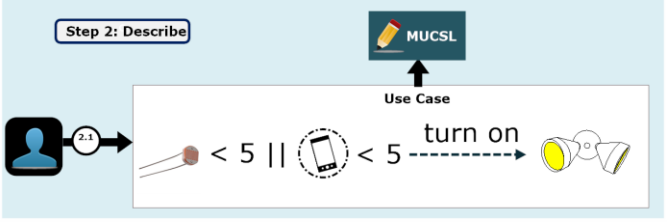



**Figure 2 Step 2: User case description by the end user**

Figure 2 depicts the Describe step using MUCSL. We can see that the user writes a use case to use the photoresistor sensor and a sensor of his smartphone. Exactly, he wants to turn on the lights when one of these sensors will have a value less than five. Clearly, as we explain before, he has to write this use case using the structure of

MUCSL.

### 2.2 Use Case Transformation

The previous section explains how the first three layers of MUCSL can support users to write a use case. This section describes how the last two layers can help transform a use case into the source code.

The **Template Use Case Layer** takes the text from the Use Case layer as input, removes irrelevant words such as 'the', 'is', 'will', 'to', and so on, and maps relevant data onto the template. In Figure 1, we can see that the obligatory words are transformed into keywords of the programming language, while the other words are settled in their corresponding place.

Finally, the **Use Case Instance Layer** generates the source code from the template. In our example, we have one sensor, the flame sensor. Then, we have created a use case in which we wrote that when the value of this sensor will be greater than '5', it will execute an alarm. The value of the flame sensor is reading every little time. In Figure 1, we can see that at that exact moment the value is '20'. Maybe, in the next petitions, this value could change or not, it will depend on the distance between the flame and the sensor. In the case of the Actuator ID, we merge the actuator ID with the Action ID to obtain the data that we must send to Midgar for doing the correct query. In this case, we have set to the action of that actuator, which has the ID '0', the value '5'. After that, we obtain the application to interconnect the different objects.

### 2.3 Midgar Platform

**Midgar** is a cloud-based IoT platform developed in a previous work [16]. This platform has been created according to have a specific platform for implementing and managing the different researches, and allows us to create in an easy way prototypes about the IoT for theses [25] and other research works. For instance, other contributions that have been implemented, used, and tested using Midgar have been [26]–[28].

Midgar offers RESTful web services for the registration and management of heterogeneous and ubiquitous objects for IoT applications. Midgar uses an XML-based message system for data exchange and allows for the interconnected objects to send or receive the data from each other, if the objects comply with the predefined communication rules [16], [26].

Midgar also possesses the basic Artificial Intelligence [29] (AI) capability [30] and its intelligence can be improved over time. This part has been facilitating the implementation and demonstration of different researches. Besides, Midgar contains an AI-based decision trees preprogrammed by Midgar users themselves. Thus, based on the input parameter, the platform can then know which object should perform what task, make a predefined decision accordingly, and then send it to the recipient object.

Once this object is registered, it can send and receive continuous messages to Midgar in a secure way [27]. These messages can serve to send data from the sensors and the status of this object to Midgar or to receive messages from Midgar about what the receiving object should do. Thus, Midgar is the brain of an entire IoT ecosystem. Then, Midgar is not a contribution but is an important part. Midgar is which manages and interconnects the objects, has the daemon programs that we created using MUCSL, which is the contribution of this paper, and keeps and manages all the information of the objects and daemon programs.

### 2.4 The Process of Using MUCSL

Using MUCSL is an integral part of the entire IoT application development process, as Figure 3 shows. The process consists of four simple steps: (1) **Register**—registering the objects to the IoT platform, (2) **Describe**—describing their interconnection in use cases, (3) **Transform**—transforming the use cases into source code, and finally (4) **Run**—executing the IoT applications through the interconnected objects.

These steps, which have been implemented as web services on **Midgar,** are described in detail as follows.

**Register:** This is the original service provided by **Midgar**. Users register their objects on the Midgar platform by providing the information of the objects in the XML format. This information includes the identifications of the objects, the identifications of the sensors attached to the objects and the actions that can be performed by the objects.

**Describe**: Users describe when and under what conditions they want to use which objects in a use case using MUCSL (Midgar Use Case Specific Language) (Figure 3 The proposed approach supports the development of interconnection software through a lifecycle process, consisting of a series of steps of Register, Describe, Transform, and Run: 3.1). The syntax of MUCSL is similar to the syntax of the English language. Here, the user uses the created DSL, MUCSL, to create the use case that describes the desired interconnection. To do this, he writes, using natural language, what he wants to create following a series of guidelines, which are shown in Figure 1. All this information is saved in an XML to send to the transform step.

**Transform:** The use case is sent to the **Transformer** (Figure 1: 3.1), which reads the use case, analyses it, and creates the corresponding source code according to the use case description (Figure 1: 3.2). In this step, the transformer receives the use case, which was defined by the user, in XML format with only the needed information but without irrelevant words or punctuation characters. The transformer parses it and creates an active process that contains how the interconnection of objects is. The transformer is a Java application, which is a part of the transformer situated in Step 3 (Figure 3) which reads the information of the use case. This information is read and inserted in a template that has the general information of the daemon: access to the databases (IP, password, queries), and the architecture and basic structures of the source code. Besides, the Java application completes the template with the information of the specific use cases which is contained and provided in the XML that was received in this layer: objects id, Server IP, which data of the objects have to compare, how to compare it (greater

than, less than, and so on), and how to use that data.

**Run:** In this step, an **Active Process** will execute the incoming message sent from the Transformer (Figure 1: 4.1). The message instructs the Active Process how to connect different registered objects. When the **transformer** creates the **active process**, the **Active Process** can be run in Midgar platform or in our own computer, because this process will send all the information to Midgar. The **Active Process** is continuing consulting the database of Midgar because this platform is which manages and keeps all the information of all the registered objects. Exactly, it is consulting the data of the objects that are presented in the use case that the user defined. Then, the **Active Process** reads the values of these objects and tests the condition. If the condition does not accomplish, the active process does not do anything. In the case that the condition is accomplished, then, the **Active Process** sends a query to Midgar to set the action that the user defined in the use case, Meanwhile, all the registered object in Midgar are sending the values of their sensors to the platform.

### 2.5 Used Software and Hardware

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

- The IoT Midgar platform (Updated to).
  - Ruby 2.5.1p57.
  - Rails 5.2.0.
  - Thin Web Server 1.7.2
  - MariaDB 10.1.29-MariaDB-6.
- MUCSL:
  - HTML 5, PHP 7.2.8, JavaScript without the use of external languages and using the standard.
- Application generator.
  - Java 10.

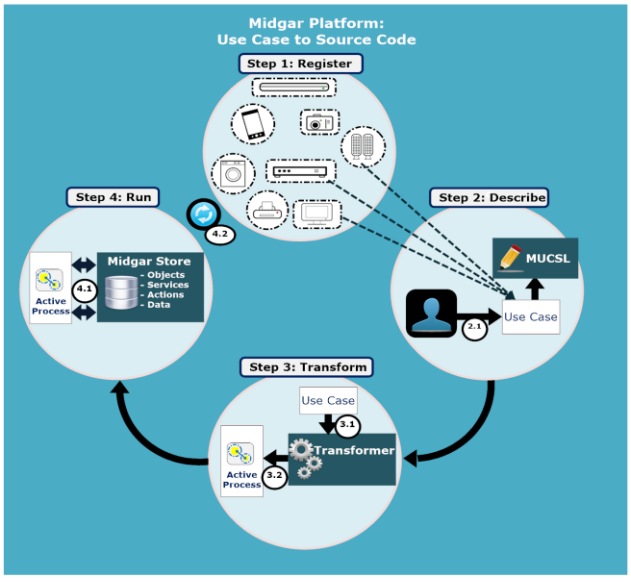



**Figure 3 The proposed approach supports the development of interconnection software through a lifecycle process, consisting of a series of steps of Register, Describe, Transform, and Run.**

Libraries used in the generated applications:

- Arduino: RXTXcomm.jar for Arduino and Java, and HTTPComponents of Apache Software Foundation.
- Android: HTTPComponents of Apache Software Foundation.

For the evaluation of the proposal, we use a server and several components that are used as Smart Objects:

- A Raspberry Pi 2 Model B as a dedicated server with Raspbian Jessie 4.9.75 v7+.
- 3 Android Smartphones and 1 emulator.
  - One Android emulator with Android 8.1.
  - Moto 5GS Plus with Android 7.1.
  - Nexus 4 with Android 5.1.1.
  - Motorola with Android 2.2.2.
- Arduino Uno SMD Microcontroller based on ATmega328.
- During the different tests and evaluations, we have used the next sensors and actuators:
  - ThermistorTMP36, a speaker, a servomotor, a DC motor, different LEDs, two buttons, a photoresistor, the temperature and humidity sensor DHT11, and the flame detector KY026.

### 2.6 Use Case Examples

In this section, we show different examples to demonstrate the capacity of the DSL:

- Combination of two sensors that have to be equal than five and another one than has to be less than 20 to activate the action that requires or needs, or may be allowed sending a not default value, which is 100 in this case. This use case could be a system to automate the heather based on temperature sensors, or a sound alarm or an automatic door that has two proximity sensors to cover range define in meters.
  - When the B8AC6F48E370-0, B8AC6F48E370-1 are equal to 5 and B8AC6F48E370-1 is less than 20 then the C3b9f28c24f2be8b-0 to 100.
- A sensor that has to be greater than 5 and a second sensor that has to be less than 20. If both cases are accomplished, then the application activates two actions, which do not need any value. This use case could be an alarm system with two actions like sound and pictures or VoIP calls or text messages, or a system to ventilate a room according to two gas sensors.
  - When the B8AC6F48E370-0 is greater than 5 and B8AC6F48E370-1 is less than 20 then the C3b9f28c24f2be8b-0 and D4az78t31y7ghu8p-0.
- When the values of the three sensors are greater than 28, then the first and second actions of the second device will be launched. In

all other cases, the actions 0 and 1 of the third device will be run. This use case could be matched with distributed sensors along a place to detect a determinate measure that could be dangerous or uncomfortable gases (CO, $CO_2$, $H_2$, etc.), or temperature, to open different windows or doors, activate an alarm, start an extractor, other ones, or various of them.

  - If B8AC6F48E370-0, B8AC6F48E370-2, B8AC6F48E370-3 are greater than 28 then the C3b9f28c24f2be8b-0, C3b9f28c24f2be8b-1 else the Z7kl94iop22ns89e-0 and Z7kl94iop22ns89e -1.

- In case we want to increment or do a repetitive task, we can use the loop to launch one or more action when one or more sensors reach the defined parameter/s.
  - Meanwhile, the B8AC6F48E370-1 is less than 10 then the Z7kl94iop22ns89e-0
- This use case can be a loop that is executing meanwhile the temperature is greater than 30ºC and the humidity is greater than 80% to call two services to increment the air conditioning and the dehumidifier until the sensors detect again a normal temperature (< 30ºC and < 80% humidity). In the other case, the data will be sent to a log web service.
  - While the B8AC6F48E370-0 is greater than 30 and the B8AC6F48E370-1 is greater than 80 then the Z7kl94iop22ns89e-1 and Z7kl94iop22ns89e-2 else Z7kl94iop22ns89e-0

The examples used in this paper are the typical ones that people use in homes or industry, but the DSL supports the use of any other device or web service that uses the Midgar platform and their systems. These devices can be RFID tags to detect when someone or something use the RFID tag, any type of sensor or actuator that you can connect to an Arduino, any sensor or actuator a smartphone has, web services, a robot that has Internet connection to move according to other sensors or devices, different smartphones, or any other device with Internet connection and the possibility of parsing an XML.

## 3 Evaluation and Results

This subsection contains the methodology used in the evaluation of this prototype and the evaluation along with its discussion. The evaluation was divided into two phases. The first phase is the taking of quantitative data from the evaluation made to the participants, which consists of three tests. The second phase is a survey made by the participants after testing the first phase, which gives us a qualitative assessment.

### 3.1 Methodology

The main objective of this evaluation has been to validate the initial hypothesis, for which the presented objectives must be fulfilled. To achieve this, MUCSL has been created, a language close to the natural language that allows, by complying with a series of rules, to transform the use cases written by the participants into the final application that will interconnect the objects.

21 participants have taken part in the evaluation:

- 95.2% had heard about the IoT.
- 38.1% had never worked with the IoT before.
- 81% had heard about Smart Objects.
- 38.1% had never worked with Smart Objects before.
- 90.5% were familiar with use cases.

Notwithstanding, the only thing that the users have to know is the rules of MUCSL. MUCSL and an introduction to the IoT, Smart Objects and the problem and solution of the research were explained before to each participant independently.

To validate the hypothesis, two phases have been developed so that the first one obtains quantitative data and the second one qualitative data to evaluate it correctly:

- Phase 1: in this first phase, three tests have been proposed to the participants. These three tests have been three possible use cases of interconnection of objects that have been written using MUCSL. This phase corresponds to the quantitative evaluation.
- Phase 2: after completing phase 1, 21 participants had to answer a survey, putting together 17 statements, using the 5-point Likert scale [31]. The survey contains statements about MUCSL and what was done in the first phase, so that the participants should state how they agree with each of the statements. This gives the qualitative data to the evaluation.

The background of the participants:

- The 95.2% of users had heard about the IoT but only the 38.1% had worked with the IoT.
- The 81% of the users had heard about Smart Objects but only the 38.1% had worked with them.
- The 90.5% had heard about use cases and the 81% had worked with them.

#### *3.1.1 Phase 1*

In this first phase, the participants had to perform three tasks using MUCSL, each task being independent of the others. The three tasks were sent to the user to facilitate understanding, being the first task easier and the second and third similar, although more difficult.

During the tests, quantitative data has been acquired regarding the use of the editors. For example, the time in seconds that each participant needed to make the application, the displacement, measured in centimetres, with the mouse and the clicks with the right and left mouse buttons. To measure this data the tool Mousotron [32] has been used.

To create the three tasks, we though in examples that can be used thoroughly by almost all people in normal and

typical tasks. However, as we have seen in section 2.6, we can use this DSL to more complex tasks.

The **first task** has been to create an application that interconnects an Arduino with a smartphone. The purpose is for this application to use the Arduino's Flame Sensor to detect fire. This first example is based on the common use of a fire detector for Smart Homes, in the Industrial IoT (IIoT), Smart Towns, Smart Cities, or in Smart Earth.

Thus, when it marks a value of 50 or less, the application should send a notification to the smartphone with the message 'Fire'. The sensor identifier is 'B8AC6F48E370-0' and the smartphone identifier is 'C3b9f28c24f2be8b-0'. A possible solution of the first task is shown in Source Code 1, where possible optional words of the phrase are shown in square brackets ('[' and ']').

```
If [the] B8AC6F48E370-0 [is] greater than
49 then C3b9f28c24f2be8b-0 [to] 'fire'
```

**Source Code 1. The solution for the first task**

The **second task** has been to develop an application that will interconnect the Arduino with a smartphone and a light. The application must use the Arduino photoresistor and the smartphone to detect whether the light needs to be turned on. This second example is based on the turn on/off the lights of a place automatically using a permanent Arduino with a photoresistor and using the smartphone like a mobile sensor. Especially, this application can be used in Smart Homes and in the Industrial IoT (IIoT) because they are indoor places.

If the Arduino sensor drops to a value of 30 or less and the smartphone to a value of 20 or less, the application must send the command to turn on the light, otherwise, the light must be turned off. The identifiers of the Arduino and smartphone sensors are 'B8AC6F48E370-0' and 'C3b9f28c24f2be8b-0' respectively. Meanwhile, the identifiers of the actions for turning on and off the light are 'D4az78t31y7ghu8p-0' and 'D4az78t31y7ghu8p-1' for each case. A possible solution to the second task is shown in Source Code 2.

```
When [the] B8AC6F48E370-0 [is] equal or
less than 30 and C3b9f28c24f2be8b-0 [is]
equal or less than 20 then [the]
D4az78t31y7ghu8p-0 else [the]
D4az78t31y7ghu8p-1
```

**Source Code 2 The solution for the second task**

In the **third task,** the participants had to develop an application that interconnects the Arduino and a fan. Thus, by using the Arduino temperature sensor the application will be able to change the fan speed of movement. This last task is the classical example based on the use of a temperature system to automate a fan in Smart Homes.

If the Arduino sensor reaches a value of 25°C or more, the application will add a speed point to the fan for each degree of difference. Otherwise, the application will turn off the fan. The identifiers are 'B8AC6F48E370-1', 'Z7kl94iop22ns89e-0' and 'Z7kl94iop22ns89e-1' for the Arduino sensor and the increase and decrease of the speed of the fan. A possible solution to the third task is shown in Source Code 3.

```
For [the] B8AC6F48E370-1 [is] greater or
equal than 25 then [the] Z7kl94iop22ns89e-
0 else [the] Z7kl94iop22ns89e-1
```

**Source Code 3 The solution for the third task**

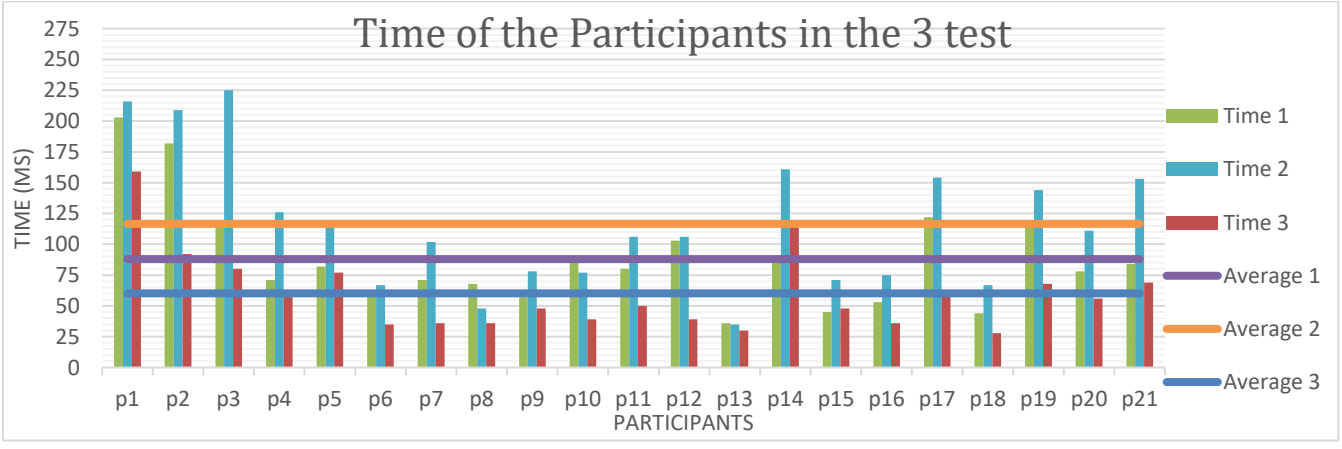



**Figure 4 Time required by each participant in each task along with the overall average**

### *3.1.2 Phase 2*

In the second phase, the qualitative part of MUCSL was evaluated to obtain the opinion of the participants and to know what they think on this research. To do this survey, the 5-point Likert scale has been used as an evaluation method because it is a widely used method in the field of software engineering to obtain information effectively to support decision making [33].

When using the 5-point Likert scale, containing a total of 17 statements, 5 possible responses are offered: 1 as Totally Disagree, 2 as Disagree, 3 as Neutral, 4 as Agree, and 5 as Totally agree.

The participants always performed this survey after completing phase 1, anonymously and without help. This survey contains statements about MUCSL, its possibilities and possible impact on the Internet of Things and Smart Objects, offering an interconnection between the two.

Table 2 contains the statements.

| # | Declarations |
|---|---|
| D1 | You can understand the structure of the elements and their role in the application creation process. |
| D2 | This tool offers a useful help to interconnect heterogeneous objects. |
| D3 | The syntax is clear, easy, and natural. |
| D4 | This solution offers a fast way for you to specify how you wish to connect your devices. |
| D5 | This solution provides assistance to interconnect objects. |
| D6 | The way to create interconnections using this language is understandable. |
| D7 | The language does not require the user to use complex programming skills, as in traditional application development. |
| D8 | The syntax includes enough elements and functionality for the user to create a wide range of interconnections to objects. |
| D9 | This proposal is a positive contribution to encourage the development of services and applications for the Internet of Things. |
| D10 | Internet of Things and Smart Objects will benefit from this solution. |
| D11 | This language could be used to simplify the classic development process of software applications in other areas (education, video games, and so on). |
| D12 | The use of this language reduces the complex development for this type of applications. |
| D13 | This syntax provides an easy and intuitive way to interconnect devices |
| D14 | The syntax of the use cases and the role of the use cases in application creation process are clear. |
| D15 | The user makes less mistakes if he uses this language than if they programmes |
| D16 | The user works quicker and more effective if he uses this language than if they programmes |
| D17 | This language can be useful |

**Table 2 Statements in the MUCSL survey**

## 3.2 Results

This section analyzes and discusses the results obtained in the two phases. Section 3.2.1 provides a quantitative analysis of the results obtained by performing the three tasks with MUCSL. The qualitative results obtained from the survey are presented in Section 3.2.2.

### *3.2.1 Results of Phase 1*

In this phase, the time of each participant has been taken while performing the three tasks required. This time is shown in Figure 4, which contains the time of each participant in each task and the overall average among all participants to perform each task.

Analyzing this graph, the following interpretations can be suggested:

- All participants took a little longer to perform the second tasks compared to the first, with the exceptions of P8, P10 and P13.
- The third task, with a difficulty like the second one, took less than the two previous tasks, except P14 and p15.
- In general terms, it seems to provide a rapid learning curve due to the decrease in the average of the third task compared to the previous two. The increment in the second task seems to be due to the increase in the difficulty of the requested task.

### *3.2.2 Results of Phase 2*

Table 3 shows the responses of each participant for each survey statement.

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | D16 | D17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *P01* | 3 | 5 | 5 | 4 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 4 | 4 | 5 |
| *P02* | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 4 | 4 | 5 | 5 | 4 | 5 |
| *P03* | 3 | 4 | 3 | 4 | 4 | 3 | 5 | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 4 |
| *P04* | 4 | 3 | 5 | 3 | 3 | 5 | 5 | 3 | 5 | 4 | 4 | 4 | 5 | 5 | 4 | 4 | 5 |
| *P05* | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| *P06* | 4 | 4 | 4 | 4 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 |
| *P07* | 4 | 5 | 5 | 4 | 5 | 5 | 4 | 3 | 5 | 4 | 5 | 3 | 4 | 4 | 5 | 5 | 5 |
| *P08* | 5 | 4 | 4 | 4 | 5 | 4 | 3 | 3 | 4 | 4 | 2 | 4 | 4 | 5 | 5 | 4 | 5 |
| *P09* | 4 | 5 | 4 | 4 | 3 | 4 | 5 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 3 | 4 | 3 |
| *P10* | 5 | 5 | 4 | 5 | 5 | 4 | 3 | 4 | 5 | 4 | 2 | 4 | 2 | 5 | 4 | 5 | 5 |
| *P11* | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| *P12* | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 5 |
| *P13* | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 3 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 5 |
| *P14* | 5 | 4 | 3 | 4 | 4 | 4 | 2 | 3 | 3 | 3 | 4 | 3 | 4 | 3 | 3 | 3 | 4 |
| *P15* | 4 | 5 | 4 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 2 | 4 | 4 | 4 | 4 | 4 | 4 |
| *P16* | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 4 |
| *P17* | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | 4 | 4 |
| *P18* | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| *P19* | 5 | 4 | 4 | 5 | 4 | 5 | 4 | 4 | 4 | 4 | 3 | 3 | 5 | 5 | 3 | 4 | 4 |
| *P20* | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 |
| *P21* | 4 | 5 | 4 | 5 | 4 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 5 |

**Table 3 Participant responses for each MUCSL statement**

Table 4 contains the global descriptive statistics of the MUCSL survey evaluation. This table shows the minimum, the first quartile, the median or second quartile, the third quartile, the maximum, the range between quartiles and the mode of each of the 17 statements of the survey.

| | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 | D16 | D17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Min | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 | 3 |
| Quartile1 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 4 | 4 | 4 | 4 | 4 | 4 |
| Median | 4 | 5 | 4 | 5 | 5 | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 5 | 5 | 5 | 4 | 5 |
| Quartile3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Max | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| Range | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 3 | 2 | 2 | 2 | 2 |
| Inter. range | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| Mode | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 4 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | 4 | 5 |

**Table 4 MUCSL global descriptive statistics**

Figure 5 shows the data for each question represented in a box in the diagram.

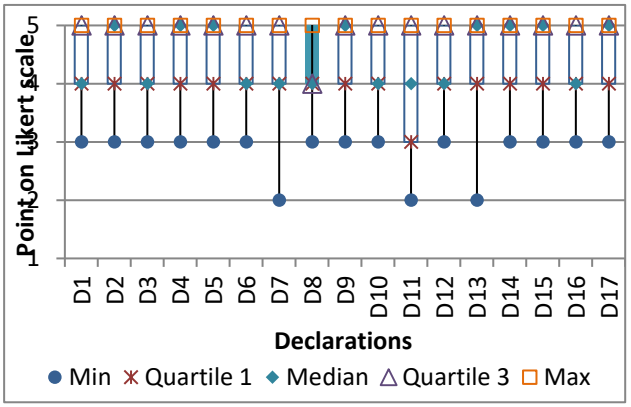



**Figure 5 Global Box Diagram for each MUCSL statement**

Thus, from the data collected and shown in Table 4 and Figure 5 the following interpretations can be suggested:

- All statements have a maximum of 5, which indicates that at least 1 person has fully agreed with each.
- All statements, except D7, D11 and D13, have a minimum of 3, indicating that in these statements the worst case has been 'neutral'.
- D7, D11 and D13 have the smallest minimum as 'disagreement'.
- D2, D4, D5, D9, D13, D14, D15 and D17 have the highest median. From this, we can deduce that most of the participants are totally in agreement with these statements.
- D7, D11 and D13 have a range of 3, which indicates that there is a great diversity of opinions on these statements. The rest of the statements have a range of 2, which indicates that the participants have a similar basic opinion on those statements.
- According to the mode, we can see that D1, D2, D4, D5, D9, D11, D12, D13, D14, D15 and D17 have a mode of 5, which indicates that most of the chosen answers have been 'totally agree'. The rest have a mode of 4, indicating that the next most chosen answer has been 'agree'.
- D11 with a range of 3, a minimum of 2, a median of 4, a mode of 5, an interquartile range of 2 and a maximum of 5, is the most dubious statement, although the most repeated answer was 'Totally agree'. On the other hand, looking at D7, which has a range of 3, a minimum of 2, a median of 4, a mode of 4, an interquartile range of 1 and a maximum of 5, is the worst-valued statement. However, D13 with a range of 3, a minimum of 2, a median of 5 a mode of 5, an interquartile range of 1 and a maximum of 5 is very well valued, although there are some participants that do not agree.
- D8 is one of the worst rated statements because it has a range of 2, a minimum of 3, a median of 4, is the only one with the quartile 3 in 4, an interquartile range of 0 and a mode of 4. This indicates that most participants are only in agreement with it and have a very similar opinion among them since it is the statement with a lower interquartile range.

Table 5 shows the different frequencies obtained from the survey for each statement based on the answers chosen by the participants. This table contains the breakdown of each question to show the number of votes for each decision and their corresponding percentage.

| Statement | | Strongly Disagree | Disagree | Neutral | Agree | Strongly Agree |
|---|---|---|---|---|---|---|
| **D1.** | # | 0 | 0 | 2 | 9 | 10 |
| | % | 0% | 0% | 10% | 43% | 48% |
| **D2.** | # | 0 | 0 | 1 | 6 | 14 |
| | % | 0% | 0% | 5% | 29% | 67% |
| **D3.** | # | 0 | 0 | 2 | 12 | 7 |
| | % | 0% | 0% | 10% | 57% | 33% |
| **D4.** | # | 0 | 0 | 1 | 9 | 11 |
| | % | 0% | 0% | 5% | 43% | 52% |
| **D5.** | # | 0 | 0 | 2 | 8 | 11 |
| | % | 0% | 0% | 10% | 38% | 52% |
| **D6.** | # | 0 | 0 | 1 | 10 | 10 |
| | % | 0% | 0% | 5% | 48% | 48% |
| **D7.** | # | 0 | 1 | 2 | 12 | 6 |
| | % | 0% | 5% | 10% | 57% | 29% |
| **D8.** | # | 0 | 0 | 5 | 12 | 4 |
| | % | 0% | 0% | 24% | 57% | 19% |
| **D9.** | # | 0 | 0 | 1 | 7 | 13 |
| | % | 0% | 0% | 5% | 33% | 62% |
| **D10.** | # | 0 | 0 | 1 | 11 | 9 |
| | % | 0% | 0% | 5% | 52% | 43% |
| **D11.** | # | 0 | 3 | 3 | 5 | 10 |
| | % | 0% | 14% | 14% | 24% | 48% |
| **D12.** | # | 0 | 0 | 3 | 9 | 9 |
| | % | 0% | 0% | 14% | 43% | 43% |
| **D13.** | # | 0 | 1 | 0 | 9 | 11 |
| | % | 0% | 5% | 0% | 43% | 52% |
| **D14.** | # | 0 | 0 | 1 | 8 | 12 |
| | % | 0% | 0% | 5% | 38% | 57% |
| **D15.** | # | 0 | 0 | 3 | 7 | 11 |
| | % | 0% | 0% | 14% | 33% | 52% |
| **D16.** | # | 0 | 0 | 1 | 10 | 10 |
| | % | 0% | 0% | 5% | 48% | 48% |
| **D17.** | # | 0 | 0 | 1 | 7 | 13 |
| | % | 0% | 0% | 5% | 33% | 62% |

**Table 5 Frequencies of the global responses of MUCSL**

Figure 6 shows a bar chart with the frequency of responses from all participants.

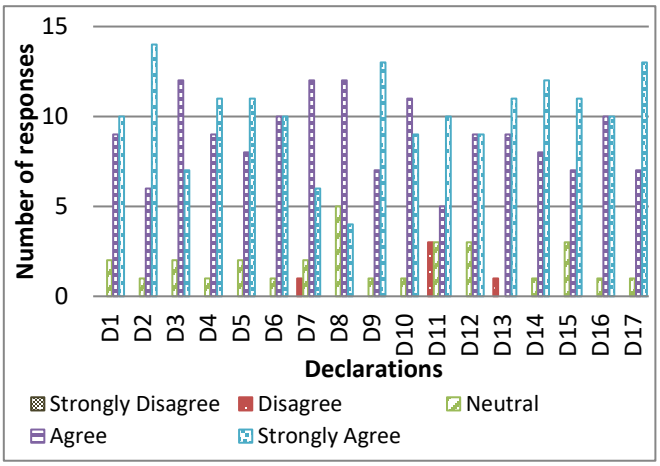



**Figure 6 Overall response distribution**

Figure 7 shows the responses of the different statements using a stacked bar graph by marking the percentiles.

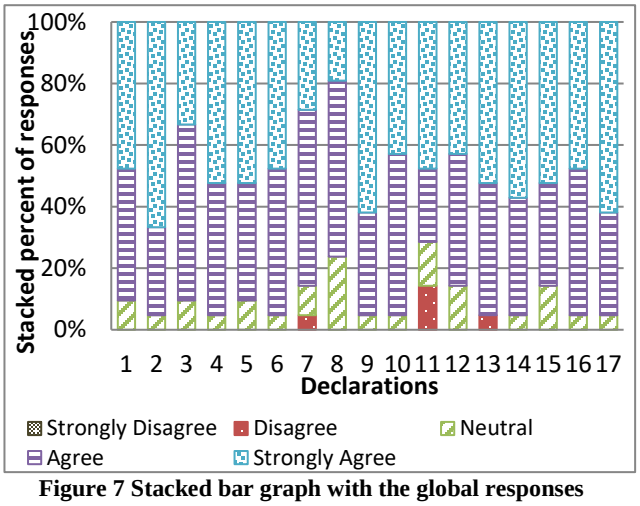



**Figure 7 Stacked bar graph with the global responses**

Based on the latest data shown, the following interpretations can be suggested:

- D2 has 67% of the votes as 'totally agree' and 29% as 'agree', while only 5% of the votes were 'neutral'. In number of votes, this is 14, 6 and 1, respectively. This indicates that the participants agree, at least in this statement, except a small minority, being this the statement best valued.
- D9 and D17 are the following two most valued statements, changing one 'totally agree' to just 'agree'.
- On the other hand, D7 is the statement with the worst opinions and with the most different responses with 57% of the votes as 'totally agree', 29% as 'agree', 10% as 'neutral' and 5% as 'disagree'. However, D11 is the worst rated by having 48% of the votes as 'totally agree', 24% as 'agree', 14% as 'neutral' and 14% as 'disagree'.

## 4 Related Work

Currently, there is still a great lack of scientific literature on DSLs in the context of this work. In fact, some works [24] emphasize that those kinds of languages are largely non-existent, and that this is one of the areas that need to be addressed in the future.

In addition, some of the few available works have different scope and objectives. For example, µPnP provides a platform-independent driver language for the IoT that enables the implementation of driver functionality in a high-level way. However, the language includes multiple options that require programming skills [34].

There are other visual languages. For example, DSL-4-IoT is a graphical modelling language, using formal presentations and abstract syntax in a metamodel to create IoT related applications [35], [36]. The PervML is another graphical DSL, created to provide developers with different elements to describe IoT systems.

Others, like the PIG DSL, focuses on a very concrete task. In this case, the definition of processes to handle the huge amount of data that is generated in the IoT in a declarative way [37].

SensApp is a platform to support cloud experiments. It uses a DSL called Gatling. It is very reminiscent of a general-purpose language due to its complexity and because of the number of possibilities it allows. For example, it is needed to create and extend classes and methods [38].

There are other works more oriented to a specific context. For example, Vitruvius [39], [40] is a platform focused on-road vehicles and is designed to simplify the collection of information in order to generate applications quickly. Therefore, interconnections are created using a DSL between vehicles and users' mobile devices with the aim of creating applications based on the collected real-time data.

In addition to the previous works, our research shows that very little current work is closely related to ours. However, for comparison purposes, we provide an overview of some representative work focusing on our previous work with Midgar, other middlewares and IoT platforms.

### 4.1 Migar platform and Migar Object Interconnection Specific Language

Domain-Specific Languages have been widely used in Web applications development such as HTML, CSS, and XML. In our early work [16], we have developed a platform and a graphical DSL for the users to specific connections between heterogeneous objects through a graphic interface. Users must define the interconnections among objects, which are registered in Midgar, using the DSL. It allows users to define conditions, actions, loops, sleeping times for objects, and even Java source code that involves any of the integrated objects. They can define relational operations using fixed numbers or using collected or real-time data. This is the way in which they must create events like 'when the temperature sensor reaches 30ºC degrees, then…' This graphic representation is sent to the parser and then automatically transformed into applications for connecting objects.

The language reported in this paper, known as Midgar-Object Interconnection Specific Language (MUCSL), complements the graphical DSL and allows the users to specify object connections in natural language, rather than in a graphical one. In this case, we also present a DSL, which is a restricted language designed specifically for object connections. However, in this case, we have developed a textual DSL, which is very close to natural language. Therefore, this facilitates the automated source code generation from use cases, as long as the follow certain rules.

### 4.2 Middleware

The Web and Internet services have been used as ubiquitous middleware to facilitate the implementation of new functionality and innovative applications for smart objects [4]. Middleware is 'a software layer or a set of sublayers interposed between the technological and the application levels' [20].

On a small scale, mashups can be thought of as a kind of middleware that uses content from more than one source to create a single new service displayed in a single graphical interface. Mashups can be used to link event and data streams from physical objects with each other as well as with Web services [4], [41].

On a larger scale, service-oriented middleware has been used to support interaction and integration between different technologies, applications or communication protocols [20]. On the Internet scale, IoT middleware has begun to emerge to facilitate communication between heterogeneous and dynamic objects at different levels of abstraction and granularity [15].

Our user-oriented language can be used to aid users to specify object connections at the application level of the IoT.

### 4.3 IoT Platforms

There are different platforms, which, although they are not specifically a DSL, they have been designed with similar objectives to those mentioned in this work.

For example, Paraimpu (Paraimpu SRL, 2012), which can integrate heterogeneous data and connect different sensors and actuators using a specific message mechanism. It allows reducing the complexity, but users still need some programming skills. In addition, it does not allow control structures. On the other hand, Midgar provides a higher level of abstractions and allows different powerful constructors like loops or timers.

Xively [42], is a supplier of cloud REST services and libraries for different platforms such as Arduino and Android. One specific conditions are met; devices can communicate with the services to perform only an action, what can reduce the possibilities of creating complex applications. Midgar, however, does not put any limitation when creating conditions and actions, and it even allows creating nested conditions.

ThingSpeak [43], is another platform to connect services and objects by creating different channels through which to transmit the data. It provides convenient mechanisms to display interactive charts. However, the system requires general-purpose programming skills to develop applications. Midgar, by contrast, is based on the use of a DSL, making it easier to perform the same type of operations

Nimbits has a different approach. It includes a downloadable server that can be used to create customized IoT servers with REST services. Midgar also uses a similar idea. One of the main differences between both is that Nimbits manage the concept of 'trigger', to perform computation depending on values of data. However, Nimbits can only use three parameters, while with Midgar there are no restrictions.

SIoT [44], [45], is a platform for the Social Internet of Things based on previous research [44]–[46]. As in other works, it allows creating channels to move data among devices. However, unlike Midgar, it lacks a graphical DSL to make it easier the creation of connections between the different smart objects involved.

Finally, Open.Sen.se that is a platform that supports several different protocols. It also uses channels to move data and create applications based on the data. It is also possible to carry out actions when different events occur. The main drawback is that the platform only allows creating simple connections among objects and Web services. In contrast, Midgar is valid to create much more complex collaboration among objects.

## 5. Conclusion and Future Work

This paper has presented a novel approach to enable end users to specify dynamic interconnections for their IoT objects to serve their own purposes. The main contributions of this approach are summarised as follows:

First, this approach contributes to our integrated development process for the IoT-based applications, which includes registering the objects to the IoT platform, describing their interconnection in use cases, transforming the use cases into source code, and finally executing the IoT applications through the interconnected objects. This approach thus addresses the challenge to create the necessary source code for the objects automatically from the use cases, provided that they keep some small rules according to the parser.

Second, our approach provides end users with a simple use case specification language MUCSL and supports automatically transformation of use cases into the source code. This allows end users to connect their own objects without the need for them to learn programming languages or operating systems of the objects. Our approach thus represents a tiny step towards bringing the IoT closer to the end users, and empowering end users with the control and connectivity of their IoT objects.

This contribution supplements deficiencies in other works previously done and provides a programming language very close to natural language to reduce the learning curve of the users at the same time it offers a clearer interface.

We have proved the concept of this approach through developing the working prototype and demonstrated the feasibility of this approach through an experiment and a

survey. The feedback from 21 participants was very positive: 19 out of 21 participants (90%) either totally agreed or agreed that our approach will be useful and beneficial for IoT. However, our approach still needs further improvements as the remaining participants (2 out 21) who commented that users would still need to have some prior knowledge of use case specification in order to use MUCSL. Despite this concern, we believe that end users can learn the basic syntax of use case specification quickly and can therefore master MUCSL easily.

Our future work will extend MUCSL to support more complex use cases so that users can specify more requirements. We will also investigate advanced NLP techniques to support the syntactic and semantic processing of the use cases. Our goal is to create a more flexible, yet powerful, language for end users. Such enhancement will enable users to describe more freely what they want from IoT applications.

Our current work has not considered the optimization, scalability, and performance of the IoT connectivity, as the number of interconnected objects is currently small. With the ever-increasing number of smart objects, their interconnections will become more complex. How to reduce the coupling of many objects while maintaining the high connectivity and performance is a major challenge in our future research.

A similar situation occurs with the security because the use of the Midgar platform offers as a secure system to send the messages that have been published in [27]. However, one of the improvements and future work that we must do is the improvement and implementation of a secure registration.

Another possible future work line could be the introduction of different protocols and data formats (JSON, CSV, etc.) to allow users to choose between the protocol and format that they want or prefer to use.

## ACKNOWLEDGEMENT

This work was performed by the 'Ingeniería Dirigida por Modelos MDE-RG' research group at the University of Oviedo under Contract No. FC-15-GRUPIN14-084 of the research project 'Ingeniería Dirigida Por Modelos MDE-RG'. Project financed by PR Proyecto Plan Regional.

## REFERENCES

[1] G. Santucci, "The internet of things: Between the revolution of the internet and the metamorphosis of objects," 2010.

[2] International Telecommunication Union, "The Internet of Things," 2005.

[3] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," 2011.

[4] F. Mattern and C. Floerkemeier, "From the Internet of Computers to the Internet of Things," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6462 LNCS, Springer, Berlin, Heidelberg, 2010, pp. 242–259.

[5] C. González García, D. Meana-Llorián, B. C. P. G-Bustelo, and J. M. C. Lovelle, "A review about Smart Objects, Sensors, and Actuators," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 4, no. 3, pp. 7–10, 2017.

[6] K. Ashton, "That 'Internet of Things' thing," *RFiD J.*, vol. 22, no. 7, pp. 97–114, 2009.

[7] G. Cueva-Fernandez, J. Pascual Espada, V. García-Díaz, and R. Gonzalez-Crespo, "Fuzzy decision method to improve the information exchange in a vehicle sensor tracking system," *Appl. Soft Comput.*, vol. 35, pp. 1–9, Oct. 2015.

[8] N. Taušan, J. Markkula, P. Kuvaja, and M. Oivo, "Choreography in the embedded systems domain: A systematic literature review," *Inf. Softw. Technol.*, vol. 91, pp. 82–101, Nov. 2017.

[9] M. Faheem and V. C. Gungor, "Energy efficient and QoS-aware routing protocol for wireless sensor network-based smart grid applications in the context of industry 4.0," *Appl. Soft Comput. J.*, 2017.

[10] B. Zeng and Y. Dong, "An improved harmony search based energy-efficient routing algorithm for wireless sensor networks," *Appl. Soft Comput. J.*, vol. 41, pp. 135–147, 2016.

[11] E. Lee, Y.-G. Kim, Y.-D. Seo, K. Seol, and D.-K. Baik, "RINGA: Design and verification of finite state machine for self-adaptive software at runtime," *Inf. Softw. Technol.*, vol. 93, no. September 2016, pp. 200–222, Jan. 2018.

[12] N. M. do Nascimento and C. J. P. de Lucena, "FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the Internet of Things," *Inf. Sci. (Ny).*, vol. 378, pp. 161–176, 2017.

[13] J. Carretero and J. D. García, "The Internet of Things: connecting the world," *Pers. Ubiquitous Comput.*, vol. 18, no. 2, pp. 445–447, Feb. 2014.

[14] Y. Sun, R. Bie, P. Thomas, and X. Cheng, "Theme issue on advances in the Internet of Things: identification, information, and knowledge," *Pers. Ubiquitous Comput.*, vol. 19, no. 7, pp. 985–987, 2015.

[15] K. Gama, L. Touseau, and D. Donsez, "Combining heterogeneous service technologies for building an Internet of Things middleware," *Comput. Commun.*, vol. 35, no. 4, pp. 405–417, Feb. 2012.

[16] C. González García, C. P. García-Bustelo, J. P. Espada, and G. Cueva-Fernandez, "Midgar: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios," *Comput. Networks*, vol. 64, no. C, pp. 143–158, Feb. 2014.

[17] Y. Zhang, Y. Xiang, X. Huang, X. Chen, and A. Alelaiwi, "A matrix-based cross-layer key establishment protocol for smart homes," *Inf. Sci. (Ny).*, vol. 429, pp. 390–405, 2018.

[18] J. I. R. Molano, J. M. C. Lovelle, C. E. Montenegro, J. J. R. Granados, and R. G. Crespo, "Metamodel for integration of Internet of Things, Social Networks, the Cloud and Industry 4.0," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–15, Feb. 2017.

[19] O. Vermesan and P. Friess, *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems*. Aalborg, Denmark: River Publishers, 2013.

[20] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[21] H. Cervantes and R. S. Hall, "Autonomous adaptation to dynamic availability using a service-oriented component model," in *Proceedings. 26th International Conference on Software Engineering*, 2004, vol. 3, no. October, pp. 614–623.

[22] S. J. Bolaños Castro, R. González Crespo, and V. H. Medina García, "Patterns of Software Development Process," *Int. J. Interact. Multimed. Artif. Intell.*, vol. 1, no. 4, p. 33, 2011.

[23] I. Jacobsson, M. Christersson, P. Jonsson, and G. Övergaard, *Object-oriented software engineering: A use-case driven approach*. Addison-Wesley, 1992.

[24] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of Internet-of-Things platforms," *Comput. Commun.*, vol. 89, pp. 5–16, 2016.

[25] C. González García, "MIDGAR: Interoperability of objects in the Internet of Things scenario using Model-Driven Engineering," *J. Ambient Intell. Smart Environ.*, vol. 9, no. 6, pp. 799–801, Nov. 2017.

[26] C. González García, J. P. Espada, E. R. N. Valdez, and V. García-Díaz, “Midgar: Domain-Specific Language to Generate Smart Objects for an Internet of Things Platform,” in *2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2014, pp. 352–357.

[27] G. Sánchez-Arias, C. González García, and B. C. Pelayo G-Bustelo, “Midgar: Study of communications security among Smart Objects using a platform of heterogeneous devices for the Internet of Things,” *Futur. Gener. Comput. Syst.*, vol. 74, no. September, pp. 444–466, 2017.

[28] D. Meana-Llorián, C. González García, B. C. Pelayo G-Bustelo, J. M. Cueva Lovelle, and N. Garcia-Fernandez, “IoFClime: The fuzzy logic and the Internet of Things to control indoor temperature regarding the outdoor ambient conditions,” *Futur. Gener. Comput. Syst.*, vol. 76, pp. 275–284, Nov. 2017.

[29] C. G. García, E. Núñez-Valdez, V. García-Díaz, C. Pelayo G-Bustelo, and J. M. Cueva-Lovelle, “A Review of Artificial Intelligence in the Internet of Things,” *Int. J. Interact. Multimed. Artif. Intell.*, vol. InPress, no. InPress, p. 1, 2018.

[30] C. González García, D. Meana-Llorián, B. C. Pelayo G-Bustelo, J. M. Cueva Lovelle, and N. Garcia-Fernandez, “Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes,” *Futur. Gener. Comput. Syst.*, vol. 76, pp. 301–313, Nov. 2017.

[31] R. Likert, “A technique for the measurement of attitudes,” *Arch. Psychol.*, vol. 22, pp. 1–55, 1932.

[32] Blacksun Software, “Mousotron,” 2016. [Online]. Available: http://www.blacksunsoftware.com/mousotron.html. [Accessed: 25-Jul-2016].

[33] M. Kasunic, *Designing an effective survey*. Pittsburgh, 2005.

[34] F. Yang, N. Matthys, R. Bachiller, S. Michiels, W. Joosen, and D. Hughes, “µPnP: plug and play peripherals for the internet of things,” in *Proceedings of the tenth European conference on computer systems*, 2015, p. 25.

[35] E. Azadi Marand, E. Azadi Marand, and M. Challenger, “DSML4CP: A Domain-specific Modeling Language for Concurrent Programming,” *Comput. Lang. Syst. Struct.*, vol. In Press, 2015.

[36] A. Salihbegovic, T. Eterovic, E. Kaljic, and S. Ribic, “Design of a domain specific language and IDE for Internet of things applications,” in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2015 38th International Convention on*, 2015, pp. 996–1001.

[37] A. F. Gates *et al.*, “Building a High-Level Dataflow System on top of Map-Reduce: The Pig Experience,” *Proc. VLDB EndowmentVldb ’09*, pp. 1–12, 2009.

[38] S. Mosser, F. Fleurey, B. Morin, F. Chauvel, A. Solberg, and I. Goutier, “Sensapp as a reference platform to support cloud experiments: From the internet of things to the internet of services,” in *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2012 14th International Symposium on*, 2012, pp. 400–406.

[39] G. Cueva-Fernandez, J. P. Espada, V. García-Díaz, C. González García, and N. Garcia-Fernandez, “Vitruvius: An expert system for vehicle sensor tracking and managing application generation,” *J. Netw. Comput. Appl.*, vol. 42, pp. 178–188, Jun. 2014.

[40] V. García-Díaz, J. P. Espada, and G. C. Fernández, “Vitruvius: Vehicle sensor based model-driven engineering application generation,” *J. Ambient Intell. Smart Environ.*, vol. 10, no. 1, 2018.

[41] D. Guinard, V. Trifa, T. Pham, and O. Liechti, “Towards physical mashups in the Web of Things,” in *2009 Sixth International Conference on Networked Sensing Systems (INSS)*, 2009, pp. 1–4.

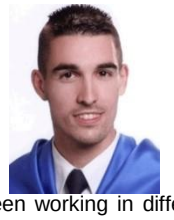
[42] LogMeIn, “Xively,” 2013. [Online]. Available: https://xively.com/. [Accessed: 29-Jul-2015].

[43] IoBridge, “Thingspeak,” 2013. [Online]. Available: http://www.thingspeak.com. [Accessed: 29-Jul-2015].

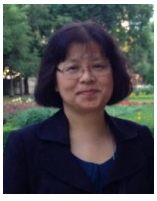
[44] L. Atzori, A. Iera, and G. Morabito, “SIoT: Giving a Social Structure to the Internet of Things,” *IEEE Commun. Lett.*, vol. 15, no. 11, pp. 1193–1195, Nov. 2011.

[45] R. Girau, M. Nitti, and L. Atzori, “Implementation of an Experimental Platform for the Social Internet of Things,” in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013, pp. 500–505.

[46] L. Atzori, A. Iera, G. Morabito, and M. Nitti, “The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization,” *Comput. Networks*, vol. 56, no. 16, pp. 3594–3608, Nov. 2012.

**Cristian González García** is a Technical Engineer in Computer Systems, M.Sc. in Web Engineering, and a Ph.D. in Computers Science graduated from School of Computer Engineering of Oviedo in 2011, 2013, and 2017 (University of Oviedo, Spain). He has been a visiting Ph.D. candidate in the University of Manchester, United Kingdom. Besides. He has been working in different national and regional projects, in projects with private companies, and in the University of Oviedo as a professor. He has published 9 journals articles, 4 conference articles, and 2 book chapters. His research interests are in the field of the Internet of Things, Web Engineering, Mobile Devices, and Modelling Software with DSL and MDE.



**Liping Zhao** is an academic member at the School of Computer Science, the University of Manchester. Her current research focuses on two areas: the application of novel natural language tools for requirements engineering and the development of software services systems to support scientific big data analytics and management. She is an associate editor for Expert Systems—The Journal of Knowledge Engineering, Wiley-Blackwell, a member of the editorial board of Requirements Engineering Journal, Springer, and a member of editorial board of Services Transactions on Cloud Computing. She is also a member of IEEE Computer Society's Technical Committee on Services Computing. She has published over 100 papers in journals and conferences.

**Vicente García-Díaz** is a Software Engineer and has a PhD in Computer Science from the University of Oviedo since 2011. He is an Associate Professor in the Department of Computer Science at the University of Oviedo. He is also part of the editorial and advisory board of several journals, and has been editor of several special issues in books and journals. He has supervised more than 60 academic projects and published more than 70 research papers in journals, conferences and books. His research interests include machine learning, natural language processing, model-driven engineering and domain specific languages.