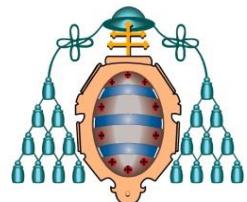


# UNIVERSIDAD DE OVIEDO



UNIVERSIDAD DE OVIEDO



## ESCUELA DE INGENIERÍA INFORMÁTICA

### TRABAJO FIN DE MÁSTER

“MIDGAR: Plataforma para la generación dinámica de aplicaciones distribuidas basadas en la integración de redes de sensores y dispositivos electrónicos IoT”

**DIRECTORES:** Dra. B. Cristina Pelayo García-Bustelo

Dr. Jordán Pascual Espada

**AUTOR:** Cristian González García



## AGRADECIMIENTOS

He de agradecer este trabajo a todas aquellas maravillosas personas que me apoyaron durante este último año, siempre, sin pedir nada a cambio. Sin ellas esto no hubiese sido posible.

A toda mi familia. Mis padres, tíos, primos, abuelos. Porque siempre estuvisteis ahí para todo lo que necesite.

A mis queridos caribeños: Andrea, Edward, Gio y Guille. Fue un año con vosotros, pasando todas las semanas a vuestro lado desde bien temprano hasta bien tarde. Gracias por vuestro apoyo y por ofrecerme vuestro tiempo y vuestra amistad. Fuisteis mi mayor apoyo este año y siempre que os necesité estuvisteis ahí. Muchas Gracias.

A Cristina, Cueva y Oscar. Por ofrecerme la oportunidad de trabajar con vosotros y, desde el primer día que empecé, haberme dado todo el apoyo que necesite para el Máster y el trabajo. Gracias por estar ahí, ayudarme y apoyarme.

A los componentes del grupo de investigación WESO: Alex, César, Nacho y Juan. Por esos grandes ratos, charlas y descansos. Por esas risas y esos consejos. Por compartir la mesa y los buenos momentos. Si conseguís lo que buscáis, se os echará en falta, pues el laboratorio ya no será nunca igual sin vosotros.

A todos los compañeros del Máster. Fueron dos años, con penas y glorias pero muy bien pasados. Muy buenos momentos, buenos cafés, buenas clases y buenas cenas. Gracias compañeros.

A los participantes de las pruebas. Gracias por venir a ayudarme. Sin vosotros, no tendría la evaluación.

A Ismael y Jordán. Gracias por vuestra ayuda. Gracias por todo. Gracias compañeros.

A Yuri. Sin ti mucho de esto no hubiera sido posible. Espero poder tener que pedirte diseños de nuevo, pues, como compañero y persona, no hay oro que te pague.

A todos mis amigos, compañeros de fútbol, pádel y compañeros de la Ingeniería. Sois muchos para nombrar, pero os veo a menudo, casi todas las semanas. Sabéis quienes sois. Siempre estuvisteis y estáis ahí. Lleváis muchos años a mi lado y espero que sigáis por muchos más. Aunque suene avaricioso, espero me sigáis cediendo vuestro valioso tiempo, pues, se os quiere.

A todos los profesores que me disteis clase, tanto en la Ingeniería como Máster. De alguna manera u otra, aprendí algo de todos vosotros. Gracias por vuestra dedicación.

## RESUMEN

Hay dos ideas que describen el futuro: Smart Objects e Internet of Things. Este pasa por la interconexión de objetos para extender su inteligencia o dotarlos de una propia. Para ello, se hace uso de la conexión a una red de comunicación que los conecte.

Tres conceptos que sirven para ilustrar esto son Smart Home, Smart City y Smart Earth. Estos conceptos se basan en la conexión de sensores y actores para dotar de “inteligencia” a las casas, ciudades y al entorno que nos rodea. El objetivo principal de estos es facilitar la vida. No obstante, se necesita una red que consiga “entender” a estos objetos y los comunique entre sí. Una red que se prevé que en un futuro tenga más tráfico de datos pertenecientes a objetos que a personas.

Esta red permitirá que se puedan desarrollar aplicaciones para dotar de esa “inteligencia” a diferentes objetos cotidianos: teléfonos, tabletas, televisiones, botellas, comida, neveras, frigoríficos, gafas, relojes, bombillas, armarios, ropa, puertas, ventanas, casas, coches, trenes, barcos, presas, puentes, carreteras,... U obtener datos de ciertos lugares remotos en busca de información útil: bosques, lagos, mares, océanos,... Con esto, se podrá: hacer que el ventilador de la habitación se encienda cuando hace calor; llamar a los bomberos cuando un bosque pueda comenzar a arder; llamar a los técnicos de reparación cuando una presa o un puente comience a envejecer y necesite revisión; tener coches llamando a la ambulancia tras un accidente y avisar a los coches próximos de un peligro en la carretera debido al accidente; monitorizar pacientes enfermos; ahorrar electricidad y agua en base a ciertos eventos...

Esto será parte del futuro. Pero para ello primero hay que resolver varios problemas fundamentales: Los objetos inteligentes o dispositivos electrónicos son a menudo muy distintos entre sí, diferentes protocolos de comunicación, esquemas de comunicación, diferente paso de mensajes, etc. Por eso no resulta del todo fácil que los objetos se comuniquen y trabajen de forma conjunta. Por eso, hay que tratar de resolver dicha heterogeneidad de los diferentes sensores y dispositivos para que se comporten como uno; Por otro lado, las personas pueden tener interés en definir cómo deben trabajar y coordinarse los objetos inteligentes, sobre todo si se trata de su casa o su lugar de trabajo. Las diferentes necesidades de cada persona pueden requerir una personalización en la coordinación entre objetos inteligentes y también puede que exista la necesidad de adaptarse de manera sencilla a futuros cambios, como puede ser un nuevo electrodoméstico en casa.

Por ello, en esta investigación se busca el permitir la interconexión de estos objetos a cualquier persona que lo desee y lo necesite sin necesidad de conocimientos informáticos de una manera rápida y sencilla. No obstante, para ello primero hay que conseguir crear una red que consiga soportar la interconexión de objetos heterogéneos y sea capaz de mantener la conexión entre estos. La solución que se propone es, mediante la utilización de Ingeniería Dirigida por Modelos, ofrecer la posibilidad a un usuario no experimentado crea desarrollar aplicaciones que interconecten objetos de una manera rápida y sencilla. Para ello se proponen dos posibles soluciones: un Lenguaje de Dominio Específico capaz de abstraer el problema de la generación de aplicaciones y un editor gráfico que facilite la creación de las aplicaciones a los usuarios sin experiencia.

## **PALABRAS CLAVE**

Internet de las Cosas; Computación UbiCua; Redes de sensores; Objeto Inteligente; Ingeniería Dirigida por Modelos; MDE; Lenguaje de Dominio Específico; DSL;

## ABSTRACT

There are two ideas that describe the future: Smart Objects e Internet of Things. This passes through the interconnection of objects to extend its intelligence or expand their intelligence if they already had it. All this thanks the use of a connection with a communication network that connects them.

Three concepts that serve to illustrate all this are Smart Home, Smart City and Smart Earth. These concepts are based in the sensor and actor connection to provide houses, cities and the environment with "intelligence". The principal objective is an easier life. But it is needed a network that achieves to "understand" these objects and communicate these object with others. A network that plans in the future have more object's data traffic than persons' data.

This network will allow you to develop applications that provide "intelligence" to different everyday objects: phones, tablets, televisions, bottles, food, fridges, refrigerators, eyeglasses, watches, bulbs, cupboards, wear, doors, windows, houses, cars, trains, boats, dams, bridges, roads,... Or getting data of some faraway places: forests, lakes, seas, oceans,... With this can make, for example: that the air vent in the room turn on when it is hot; notifying the firemen when a forest can begin to burn; call the maintenance engineer when a dam or a bridge start to get old and they need to be repaired; keep calling the ambulance car after an accident and alert the next car of a hazard on the road due to accident; monitoring of ill people; notifying the traffic conditions in the roads or free parking in a place; save electricity and water according to certain event...

This will be part of the future. But to do this you first have to solve several fundamental problems: smart object or electronic devices are often very different each other, different protocols of communication, different communication schema, different ways of messages, etc... It is for that it isn't easy that objects communicate among them and work in a same way. This heterogeneity of different sensors and devices has to be solved for working like one alone. On the other hand, people can have an interest in defining the work and the coordination of Smart Objects, especially if it is about their houses or their work places. The different needs of each person may require customization coordination between Smart Objects and also a need to be able to adapt for future changes, for example, an electrical household appliance at home.

Therefore, this research seeks to allow interconnection of these objects for any person who want it and need it without any computer knowledge and in a fast and easy way. Nevertheless, first we have to get a network that supports the interconnection of heterogeneous objects and can keep the connection among them. The proposed solution is that an ordinary user can develop some applications for interconnecting objects in a fast and easy way, thanks the use of Model-driven engineering. To do this we propose two possible solutions: a Domain Specific Language can abstract the problem of application generation and a graphic editor that facilitates the creation of applications to users without experience.

## **KEYWORDS**

Internet of Things; Ubiquitous Computing; Sensors Network; Smart Object; Model-driven engineering; MDE; Domain Specific Language; DSL;

## TABLA DE CONTENIDO

Agradecimientos .....	I
Resumen .....	II
Palabras Clave .....	III
Abstract.....	IV
Keywords.....	V
Tabla de contenido.....	VI
Índice de Ilustraciones .....	IX
Índice de Tablas.....	XI
1    Introducción .....	- 1 -
2    Problemas.....	- 4 -
2.1    Heterogeneidad de los objetos .....	- 4 -
2.2    Desarrollo de aplicaciones para interconectar objetos por personas inexpertas .....	- 5 -
3    Contribución.....	- 7 -
4    Posibles ámbitos de aplicación.....	- 7 -
5    Estado del arte .....	- 9 -
5.1    Internet of Things .....	- 9 -
5.2    Smart Objects .....	- 10 -
5.3    Ingeniería Dirigida por Modelos .....	- 11 -
5.4    Arquitectura Dirigida por Modelos.....	- 13 -
5.5    Lenguajes de Dominio Específico .....	- 15 -
5.6    eXtensible Markup Language.....	- 15 -
5.7    XML Metadata Interchange.....	- 16 -
5.8    XML Schema.....	- 16 -
5.9    XSLT .....	- 16 -
5.10    Plataformas IoT .....	- 17 -
5.10.1    Paraimpu .....	- 17 -
5.10.2    Xively .....	- 18 -
5.10.3    ThingSpeak .....	- 19 -

6	Caso de estudio: MIDGAR .....	- 21 -
6.1	Generación de aplicaciones interconectores de objetos heterogéneos .....	- 21 -
6.2	Arquitectura propuesta .....	- 22 -
6.3	Modelo obtenido aplicando Ingeniería Dirigida por Modelos .....	- 23 -
6.4	Implementación .....	- 25 -
6.4.1	Editor Web Textual .....	- 26 -
6.4.2	Editor Web Gráfico .....	- 26 -
6.4.3	Lenguaje de Dominio Específico .....	- 28 -
6.4.4	Service Generation .....	- 30 -
6.4.5	Processor and Object Manager.....	- 31 -
6.4.6	Objects .....	- 32 -
6.5	Software utilizado.....	- 34 -
7	Evaluación y discusión .....	- 35 -
7.1	Metodología.....	- 35 -
7.1.1	Toma de datos .....	- 35 -
7.1.2	Encuesta .....	- 37 -
7.2	Resultados de la toma de tiempos.....	- 40 -
7.2.1	Comparativa de Tiempo .....	- 44 -
7.2.2	Comparativa de pulsaciones de teclado.....	- 44 -
7.2.3	Comparativa de errores .....	- 45 -
7.2.4	Comparativa de consultas.....	- 46 -
7.3	Resultados de la encuesta .....	- 47 -
7.3.1	Evaluación global.....	- 48 -
7.3.2	Evaluación de los desarrolladores de software.....	- 51 -
7.3.3	Evaluación de los usuarios interesados en Internet of Things .....	- 55 -
8	Conclusiones .....	- 60 -
9	Trabajo Futuro.....	- 61 -
10	Difusión de los resultados .....	- 63 -
10.1	MIDGAR: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios .....	- 63 -

10.2	Using Model-driven architecture principles to generate applications based on interconnecting smart objects and sensors.....	- 66 -
10.3	Domain Specific Language for the development of Educative Multiplatform Videogames Case study GADE4ALL.....	- 67 -
11	Referencias.....	- 71 -
12	Anexo I: Acrónimos.....	- 76 -
13	Anexo II: Glosario .....	- 78 -
14	Anexo III: Índice alfabético .....	- 83 -
15	.....	- 83 -
16	Anexo IV: Contenido del DVD-Rom .....	- 85 -
16.1	Prototipos.....	- 85 -
16.2	Artículos .....	- 85 -
17	Anexo V: Artículos .....	- 86 -
17.1	MIDGAR: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios .....	- 86 -
17.2	Using Model-driven architecture principles to generate applications based on interconnecting smart objects and sensors.....	- 113 -
17.3	Domain Specific Language for the development of Educative Multiplatform Videogames Case study GADE4ALL.....	- 136 -

## ÍNDICE DE ILUSTRACIONES

Ilustración 1 Sintaxis abstracta de Ecore	- 13 -
Ilustración 2 Ejemplo de MDA distribuido por capas según el OMG	- 14 -
Ilustración 3 Logotipo de Paraimpu	- 17 -
Ilustración 4 Logotipo de Xively	- 18 -
Ilustración 5 Creación de un disparador en Xively	- 18 -
Ilustración 6 Logotipo de ThingSpeak	- 19 -
Ilustración 7 ThingSpeak: Datos en formato XML	- 19 -
Ilustración 8 ThingSpeak: Datos en formato CSV	- 19 -
Ilustración 9 ThingSpeak: Datos en formato JSON	- 20 -
Ilustración 10 Gráfica generada por ThingSpeak con información sobre la luz	- 20 -
Ilustración 11 Arquitectura de la plataforma MIDGAR	- 23 -
Ilustración 14 Metamodelo	- 24 -
Ilustración 12 Editor textual para la generación de aplicaciones interconectoras de objetos de la plataforma MIDGAR	- 26 -
Ilustración 13 Editor gráfico para la generación de aplicaciones interconectoras de objetos de la plataforma MIDGAR	- 27 -
Ilustración 15 Funcionamiento interno del Service Generation	- 31 -
Ilustración 16 Foto del Arduino durante una de las pruebas	- 34 -
Ilustración 17 Ejemplo de un diagrama de cajas y bigotes	- 40 -
Ilustración 18 Comparativa de tiempos medio de creación de las aplicaciones	- 44 -
Ilustración 19 Comparativa con las pulsaciones de teclado medias en la creación de las aplicaciones	- 45 -
Ilustración 20 Comparativa con el número de errores medios en la creación de las aplicaciones	- 46 -
Ilustración 21 Comparativa con el número de consultas medias de los usuarios en la creación de las aplicaciones	- 47 -
Ilustración 22 Diagrama de cajas y bigotes global para cada pregunta	- 49 -
Ilustración 23 Distribución de las respuestas globales	- 50 -
Ilustración 24 Distribución apilada de las respuestas globales	- 51 -
Ilustración 25 Diagrama de cajas y bigotes de los desarrolladores de software para cada pregunta	- 52 -

Ilustración 26 Distribución de las respuestas de los desarrolladores de software	- 54 -
Ilustración 27 Distribución apilada de las respuestas de los desarrolladores de software	- 54 -
Ilustración 28 Diagrama de cajas y bigotes de los IoTU	- 56 -
Ilustración 29 Distribución de las respuestas de los IoTU	- 58 -
Ilustración 30 Distribución apilada de las respuestas de los IoTU	- 58 -
Ilustración 31 Computer Communications	- 63 -
Ilustración 32 Universal Access in the Information Society	- 64 -
Ilustración 33 Computer Networks	- 64 -
Ilustración 34 Journal of Network and Computer Applications	- 65 -
Ilustración 35 Personal and Ubiquitous Computing	- 66 -
Ilustración 36 Advances and Applications in Model-Driven Engineering	- 67 -
Ilustración 37 Computers & Education	- 68 -
Ilustración 38 Computers Applications in Engineering Education	- 68 -
Ilustración 39 Educational Technology & Society	- 69 -
Ilustración 40 Journal of Science Education and Technology	- 70 -

## ÍNDICE DE TABLAS

Tabla 1 Correspondencia entre la Sintaxis Abstracta y la Sintaxis Concreta.....	- 25 -
Tabla 2 Cuestionario realizado a los usuarios.....	- 38 -
Tabla 3 Toma de datos en la realización de la aplicación con el Notepad++ .....	- 41 -
Tabla 4 Toma de datos en la realización de la aplicación con el Editor Textual .....	- 42 -
Tabla 5 Toma de datos en la realización de la aplicación con el Editor Gráfico .....	- 43 -
Tabla 6 Respuestas globales de los usuarios en cada pregunta .....	- 48 -
Tabla 7 Tabla con las estadísticas descriptivas globales.....	- 48 -
Tabla 8 Tabla de frecuencias de las respuestas globales.....	- 50 -
Tabla 9 Estadísticas descriptivas de los desarrolladores de software .....	- 52 -
Tabla 10 Tabla de frecuencias de las respuesta de los desarrolladores de software .....	- 53 -
Tabla 11 Estadísticas descriptivas de los IoTU.....	- 55 -
Tabla 12 Tabla de frecuencias de las respuesta de los IoTU .....	- 57 -

## Introducción

---

### 1 INTRODUCCIÓN

Internet of Things (IoT) es un concepto que abarca un conjunto de tecnologías que buscan permitir la interconexión e interoperabilidad de objetos heterogéneos y ubicuos a través de diferentes redes. IoT abre la puerta a muchas posibilidades, como puede ser el campo de la inteligencia ambiental (Gu & Wang, 2009), mejor conocido como Smart Earth (Hao, Lei, & Yan, 2012), las Smart Cities (Hao et al., 2012) o las Smart Home (Gu & Wang, 2009; Han, Jornet, Fadel, & Akyildiz, 2013; Hribernik, Ghrairi, Hans, & Thoben, 2011). Esto se debe a que actualmente hay un gran auge en el uso de las nuevas tecnologías como son los ordenadores, Internet, la computación en la nube, los Smartphones (Telefónica, 2013), las Tablets, los micro-controladores (Georgitzikis, Akribopoulos, & Chatzigiannakis, 2012; Piras, Carboni, Pintus, & Features, 2012) y las etiquetas inteligentes (Kortuem, Kawsar, Fitton, & Sundramoorthy, 2010; Tan, 2010). Además, hubo un gran abaratamiento de los sensores y la mejora de las redes inalámbricas (Akyildiz, 2002). Todo este conjunto de objetos y nuevas tecnologías explica que Internet of Things esté ganando mucha importancia en las telecomunicaciones modernas (Atzori, Iera, & Morabito, 2010).

IoT persigue como objetivo final tener sensores u objetos dispersos para que nos den información desde cualquier lugar del mundo. Puede que estos se encuentren en nuestra casa, en un sitio accesible o bien en el interior de una máquina o un motor. Otros, posiblemente, sean utilizados para detectar fenómenos meteorológicos y puede que estén en lo profundo del océano, en una habitación sellada con productos químicos, dentro de una central nuclear, se encuentren en medio de las filas enemigas en medio del campo de batalla o incluso, en lugares en los que una vez sean colocados, sean totalmente inaccesibles para ser recuperados (Akyildiz, 2002; Bulusu, Estrin, & Girod, 2001). Con esto se puede ofrecer la interconexión de estos objetos heterogéneos a través de una red de comunicación, como puede ser Internet (Lee & Kim, 2012). Estos objetos pueden ser sensores dispersos por el terreno (Akyildiz, 2002), un ordenador (Roman, Zhou, & Lopez, 2013), una red de sensores (Akyildiz, 2002), una micro-controlador Arduino, una Smart TV, un Smartphone, una etiqueta inteligente Radio Frequency ID (RFID), una nevera... Pues, Internet of Things es la interconexión de objetos heterogéneos a través de Internet (Atzori, Iera, Morabito, & Nitti, 2012; Hribernik et al., 2011; Kortuem et al., 2010; Lee & Kim, 2012). Esto llevará a un futuro en el que no sólo sea usado para la comunicación entre personas, si no, entre humano y máquina, e incluso, entre diferentes máquinas (M2M) (Tan, 2010). Se prevé que Internet of Things produzca un gran impacto en el cambio de vida diario, tanto en el ámbito doméstico como en el comercial (Atzori et al., 2010). Pues puede aumentar la calidad de vida como mejorar la producción de una empresa, todo mediante la interconexión de objetos y las posibilidades que ello conlleva.

Cabe destacar, que Internet of Things está ganando tanta importancia que incluso el United States National Intelligence Council la considera como una de las seis tecnologías con mayor impacto en los intereses de los Estados Unidos hasta 2025 (The US National Intelligence Council, 2008). Por otra parte, la Organización de las Naciones Unidas (ONU), en la reunión de Túnez en el año 2005, dio notoriedad a

## Introducción

---

Internet of Things, prediciendo una nueva era de ubicuidad en donde el tráfico de Internet será pequeño en comparación con el de objetos cotidianos (Atzori et al., 2010).

Por ello cobran también mucha importancia los Smart Objects: objetos físicos con un sistema embebido que le permite procesar información y comunicarse con otros dispositivos y realizar acciones en base a una acción o evento determinado (Hribernik et al., 2011). Estos pueden ser los Smartphones, un micro-controlador como es Arduino (Georgitzikis et al., 2012; Hribernik et al., 2011; Piras et al., 2012) o cualquier otro objeto que tenga conectividad a la red (Lee & Kim, 2012) y sea capaz, como mínimo, de gestionar información (Meyer, Främling, & Holmström, 2009). Pues, en combinación con Internet of Things, se consigue que la red pase de solo transportar datos, a dotarla de inteligencia y acciones según los datos que recogen estos objetos y los servicios que estos permiten realizar.

Por ejemplo, (Gu & Wang, 2009; Rothensee, 2007) hablan de una nevera que analiza los hábitos alimenticios y que puede ser muy útil para gente con enfermedades o para mejorar la calidad de vida de una familia. (Hribernik et al., 2011) investigó un sistema inteligente para dotar de cierta inteligencia a la casa y otro para manejar una carretilla elevadora según ciertas condiciones que se iban dando en los sensores que esta poseía. También esto puede ayudar a la ecología y a la economía, pues puede contribuir al ahorro energético en un campus por medio del uso de etiquetas inteligentes RFID como propuso (Tan, 2010) o bien en una casa (Lee & Kim, 2012).

Un ejemplo más ambicioso es la conexión de sensores para el monitoreo ambiental. Este podría ser crucial para la prevención o control de cierto tipo de circunstancias en las que podrían jugar un papel crucial (Broring, Maue, Malewski, & Janowicz, 2012). Por ejemplo, en el desastre del Deepwater Horizon en 2010 en el Golfo de México se podría haber utilizado para ayudar a controlar el vertido de petróleo, analizar el grado de salinidad del agua y el estado de las diferentes partes para evitar problemas mayores (Broring et al., 2012), o incluso, previamente, ayudar a prevenir ciertos tipos de desastres de este tipo mediante la monitorización de diferentes estructuras y condiciones.

No obstante, todos estos sistemas son muy complejos. El principal problema a la hora de interconectar los Smart Objects para crear una red inteligente es el trato de los diferentes valores y tipos de mensajes que envía cada dispositivo. Esto es debido tanto al diferente hardware utilizado por cada uno como al software que utiliza. Esto se debe a las normas en su creación por parte de los diferentes fabricantes (Gama, Touseau, & Donsez, 2012).

Debido a estos problemas, una persona sin conocimientos informáticos puede no ser capaz de realizar la coordinación entre objetos inteligentes de forma fácil y sin complicaciones, sobretodo, cuando los objetos sean muy diferentes técnicamente entre sí. Esto se debe a que tendría que realizar aplicaciones específicas que tratasen las diversas situaciones así como los diferentes tipos de mensajes o bien, la heterogeneidad encontrada en el hardware y software (Georgitzikis et al., 2012). Esto sería un sistema

## Introducción

---

muy costoso, tanto en dinero como en tiempo, repetitivo y que puede desembocar en errores (Dijkstra, 1972; Kent, 2002).

Por ello, el objetivo de esta investigación es, mediante la aplicación de los conceptos de la Ingeniería Dirigida por Modelos (MDE), facilitar la creación de aplicaciones que permitan interconectar objetos heterogéneos a usuarios no expertos. Para ello se diseñó un Lenguaje de Dominio Específico (DSL). Además se proporcionará un editor gráfico para facilitar la definición de las aplicaciones. Este editor implementará el DSL y permitirá la definición rápida y sencilla de aplicaciones para interconectar objetos heterogéneos. Estos podrán ser sensores, redes de sensores, objetos o Smart Objects. Se pretende conseguir que las personas que deseen crear conectar varios objetos no necesiten de conocimientos de programación. Gracias a este DSL se otorga de una abstracción al usuario respecto al lenguaje de programación base. Así, el usuario solo deberá conocer el dominio del problema y las propiedades de este, pues, el editor le facilitará el trabajo de escritura. No obstante, esta investigación implica el desarrollo de una infraestructura que de soporte a la comunicación entre los diferentes objetos. La plataforma se encargará de la centralización de todos los datos de los diferentes sensores y objetos permitiendo la interacción y comunicación entre ellos por medio de la creación de una red ubicua y resolviendo el problema de la interconexión.

## 2 PROBLEMAS

El objetivo final de Internet of Things y los Smart Objects es el poder realizar una red inteligente que controle casi cualquier cosa. No obstante, existe un gran problema en común: la heterogeneidad de los objetos. Esta heterogeneidad dificulta la comunicación entre los objetos (Gama et al., 2012). Una de las soluciones actualmente más populares, para conectar objetos es la realización de una interfaz en común. Esta puede ser una aplicación que resida en ambos y consulte un servidor. Como ejemplo a esto son los programas multiplataforma existentes, desde servicios de mensajería como son WhatsApp, Skype y Line, hasta los videojuegos multijugador, por ejemplo, Triviados y Apalabradados. No obstante, estos solo consiguen conectar los Smart Object para los que realizan su aplicación. Para esto necesitan de desarrolladores para realizarla. En cambio, el poder de IoT se encuentra realmente en el soporte para objetos heterogéneos. Sin importar el sistema operativo, plataforma o las funcionalidades. Sin importar si son inteligentes o no. Objetos domésticos o sensores dispersos por el mundo. Para conseguir esto, se necesita, no un grupo de desarrolladores, sino a toda la gente. Gente sin conocimientos de programación. Gente con deseo de conectar dichos objetos para poder beneficiarse de la tecnología y ayudar a crear un entorno mejor. Por eso, estas personas solo deberían de conocer el dominio del problema y no deberían tener que aprender a programar. Por ello, una solución a este problema sería crear un lenguaje de dominio específico para conseguir dicha capa de abstracción y encapsular el dominio del problema en este (Arie Van Deursen, Klint, & Visser, 2000).

### 2.1 *Heterogeneidad de los objetos*

Los principales problemas en Internet of Things y en las redes de sensores son muy similares debido a la naturaleza y finalidad que ambas comparten. Ambas se componen de objetos que deben conectar, en el segundo caso, solo sensores. El problema en este punto reside en todas las diferentes implementaciones que hay que realizar para los diferentes tipos de sensores en los diferentes sistemas operativos existentes. Desde el punto de vista del software, lo que puede diferir en cada caso es el tipo de mensaje que se envía. Así, los mensajes enviados entre unos Smart Objects creados por una empresa pueden diferir en gran medida de los Smart Objects de otra. Esto provoca que los objetos inteligentes de diferentes empresas no puedan interactuar entre ellos debido a la falta de entendimiento (Gama et al., 2012). Por ello una posible solución a estos problemas sería el uso de una red que consiga interconectar los diferentes dispositivos (Tan, 2010). Esta puede tratar los diferentes datos recibidos y contrastar todo bajo un mismo formato y dotar de la inteligencia de entendimiento necesaria a cada dispositivo a través de la red. Por ello, el principal problema de estas es la heterogeneidad, el dinamismo y la evolución de su contenido (Gama et al., 2012). En esta investigación, de los tres problemas que puede provocar el contenido de la red, el que nos importa principalmente es la heterogeneidad.

## Problemas

---

Estos problemas surgen de la intención de manejar diferentes dispositivos, protocolos, sensores, objetos y aplicaciones. Cada uno realizado de una forma diferente, por diferentes fabricantes y con muy poco en común con el resto. No existe una estandarización que englobe a todos estos objetos en todos los aspectos necesarios (protocolos, servicios disponibles, tipos de datos, paso de mensajes,...). Esto implica que la comunicación directa entre ellos, en la mayoría de los casos, no sea posible debido a la falta de entendimiento por la inexistencia de interfaces y un estándar o protocolo común (Guinard & Trifa, 2009). Esta diferencia entre objetos es a lo que nos referimos con el término heterogeneidad de los objetos. Por ello hay que proporcionar una arquitectura adecuada que de soporte a una comunicación fácil a cualquier tipo de dispositivo. Un ejemplo de heterogeneidad con un Smart Object puede ser un micro-controlador Arduino (Georgitzikis et al., 2012; Hribernik et al., 2011). En este caso, la heterogeneidad de sensores es mayor debido a la gran cantidad disponible de sensores para ser conectados al micro-controlador. Cualquier dispositivo electrónico de cualquier marca puede ser conectado a este. Esto hace que cualquier Arduino pueda ser diferente a otros debido a los sensores que usa, pues dependerá de los datos que este sensor utilizado recoja. Otro ejemplo pero a gran escala de problemas de heterogeneidad en Internet of Things son las Smart City. Como se comenta (Hao et al., 2012), el primer problema a resolver es la recolección de datos, el acceso y la transmisión. Esto se debe a que en Internet of Things necesita reconocer los objetos inteligentes y mantener un flujo de mensajes constante entre los diferentes objetos. No obstante, cada implementación de diferentes redes de sensores puede presentar diferentes problemas y cada investigador opta por una solución diferente (Akyildiz, 2002). Esto da lugar a múltiples soluciones y ninguna estándar (Guinard & Trifa, 2009).

Una solución en este caso será realizar una arquitectura que soporte el paso de mensajes de los diferentes tipos de dispositivo y sea capaz de responderles. Para ello, se creará una arquitectura basada en servicios (SOA) como proponen (Atzori et al., 2010; Gama et al., 2012; Pintus, Carboni, & Piras, 2011; Pintus, Carboni, Piras, Giordano, & Elettrica, 2010) y una aplicación RESTFul que sea utilizable por cualquier objeto como proponen en (Guinard, Trifa, Pham, & Liechti, 2009).

### 2.2 *Desarrollo de aplicaciones para interconectar objetos por personas inexpertas*

El desarrollo de aplicaciones requiere de personas con experiencia en el desarrollo de software y que tengan un amplio conocimiento del dominio de los sensores y objetos a conectar, así como de los diferentes lenguajes de programación que se necesitan utilizar en las diferentes plataformas. El principal inconveniente radica en el desarrollo de aplicaciones de una manera fácil y rápida por personas sin conocimientos informáticos (Guinard & Trifa, 2009). Una solución es la necesidad de ofrecer una aplicación que les permita realizar dicha tarea o les ayude con ella. De esta manera la gente que lo deseé, podrá interconectar diferentes dispositivos y sensores que implementen la especificación desarrollada y así realizar acciones bajo las circunstancias que ellos deseen.

Problemas

---

Como solución se expondrá la creación de un editor gráfico que facilite el desarrollo de dicha aplicación utilizando un Lenguaje de Dominio Específico (DSL) para ello. Así, por medio de este DSL, se conseguirá abstraer el lenguaje de desarrollo permitiendo al usuario inexperto definirlo mediante etiquetas y atributos y sin tener que programar.

### 3 CONTRIBUCIÓN

Con esta investigación se pretende realizar contribuciones significativas que faciliten el desarrollo de aplicaciones que ofrezcan interconexión de objetos inteligentes en el marco de Internet of Things (IoT) de una forma rápida y sencilla. Para ello, se pretende comprobar que mediante la posibilidad del uso de Ingeniería Dirigida por Modelos se puede solucionar este problema. Para demostrar esto se realizó:

- **Lenguaje de Dominio Específico para la creación de aplicaciones que interconecten dispositivos heterogéneos:** Se creó un Lenguaje de Dominio Específico para poder crear aplicaciones que interconecten dispositivos heterogéneos. De esta manera se dará una mayor abstracción al problema. Esto repercute en una mayor facilidad a la hora de crear aplicaciones por parte de un usuario y en una reducción del número de errores a la hora de realizar estas aplicaciones.
- **Alta abstracción para la creación de aplicaciones mediante un editor web gráfico:** Se ofrecerá a los usuarios un editor gráfico que les ayudará a escribir el Lenguaje de Dominio Específico creado para la creación de aplicaciones. De esta manera, se espera aumentar la facilidad de escribir y generar dicho DSL, así como reducir los posibles errores que puedan surgir a la hora de realizarlo.
- **Comunicación de objetos heterogéneos ubicuos:** Se consiguen conectar cualquier objeto, sin importar la plataforma, el tipo de hardware o lenguaje en el que esté realizado mediante la arquitectura propuesta en esta investigación.

### 4 POSIBLES ÁMBITOS DE APLICACIÓN

Posibles ámbitos de esta aplicación sería el soporte para la interconexión de objetos en:

- **Casas:** Conocido como Smart Home. Cualquier persona podría conectar diferentes dispositivos de su casa y entorno a ello, poder automatizarla. Por ejemplo, podría hacer un sistema que cuando se diese una determinada temperatura, se encendiesen los ventiladores o la calefacción. Incluso, este sistema, podría automatizarlo para que sólo funcionase cuando estuviese en casa o en una habitación en concreto.
- **Negocios:** Mediante la integración de Internet of Things en estos ámbitos, se puede controlar todo el proceso de manipulación de un producto para así tener controlando toda la gestión y el ciclo de vida en la distribución de este, facilitando dicha tarea y automatizando gran parte de ella.
- **Industrias y compañías:** Una de las aplicaciones más importantes en la industria, es en temas de seguridad en el almacenaje de residuos peligrosos, en el que, por medio de sensores, pueden detectarse situaciones peligrosas y alertar a los debidos responsables. Esto

mismo se puede aplicar a diferentes tipos de almacenaje, transporte e identificación y monitorización de productos.

- **Medio ambiente:** Conocido como Smart Earth. Instalación de sensores en infraestructuras creadas por los humanos y en el propio medioambiente para poder prevenir desastres, como el Deepwater Horizon, incendios o alertar, con antelación de posibles catástrofes naturales.
- **Campus Universitarios:** En estos se puede aplicar para el ahorro de energía en las diferentes aulas con la ayuda de sensores de proximidad para encender o apagar las luces o bien, en los aparcamientos de coches y bicicletas para ayudar a los estudiantes y al profesorado a conocer el estado del aparcamiento, así como ayudar a saber la necesidad de este.
- Creación de **aplicaciones de aviso** entre Smartphones. Por ejemplo, cuando los acelerómetros de un móvil excedan de una determinada fuerza G avise a otro o realice una llamada a emergencias debido a la posibilidad de un accidente.

## 5 ESTADO DEL ARTE

Objetos interactuando entre ellos y enviándose mensajes a través de una red de comunicación. Un objeto realizando una acción debido a que, otro objeto a miles de kilómetros, le notificó cierta información relevante. Una red con inteligencia que pueda tomar decisiones en base a ciertas acciones. Esto es Internet of Things y muchos de estos objetos son Smart Objects y sensores. Utilizando este tipo de redes, cualquier persona podrá automatizar y mejorar su vida diaria, así como las ciudades comenzarán a ser inteligentes, lo que repercutirá en la mejora de vida de los propios ciudadanos. No obstante, para ello hay que resolver varios problemas, esencialmente dos. Primero, la interconexión de objetos heterogéneos para que puedan trabajar bajo una misma red, con un mismo protocolo o estándar (Guinard & Trifa, 2009). Segundo, que toda la gente sea capaz de especificar la coordinación que quieren otorgar a los objetos, pues, a día de hoy, en la mayoría de los casos, hay que tener conocimientos técnicos para llevarla a cabo. Para revolver el primer problema, se necesitaría un estándar en común que utilicen todos los fabricantes de objetos. Para el segundo, habría que enseñar los conocimientos técnicos necesarios a todas las personas. No obstante, ninguna solución es factible.

Por ello, una posible solución es que las personas puedan crear las aplicaciones para interconectar objetos sin tener que programar. Solo necesitando conocer el dominio del problema. Para conseguir este objetivo, se necesita abstraer el problema lo máximo posible. Mediante Ingeniería Dirigida por Modelos (MDE) y un Lenguaje de Dominio Específico (DSL) esto es posible. Es posible que la gente no deba programar, todo mediante el uso de un lenguaje que englobe este dominio.

### 5.1 *Internet of Things*

En la actualidad, la gran mayoría de interacción realizada por Internet es del tipo humano-humano. No obstante, cada vez más Smart Objects pueden conectarse a Internet y en un futuro, se espera que haya más Smart Objects conectados que personas (Tan, 2010). Estos pueden interactuar entre ellos, enviarse datos y realizar determinadas acciones según cierta información. Objetos heterogéneos interactuando entre ellos: Esto es Internet of Things (Atzori et al., 2010; Gama et al., 2012; Hribernik et al., 2011; Kortuem et al., 2010; Lee & Kim, 2012; Tan, 2010). IoT surgió en base a la necesidad de las cadenas de suministro y la identificación de objetos, personas y animales mediante el uso de etiquetas inteligentes RFID (Atzori et al., 2010; Gama et al., 2012; Kortuem et al., 2010). Esto se debe a que con el uso de RFID se puede otorgar un identificar único a un objeto (Atzori et al., 2010). De esta manera, se puede conseguir localizar a un objeto concreto para determinadas tareas. No obstante, como explica (Tan, 2010), para la existencia de Internet of Things, se necesitan tres pasos: inteligencia integrada, conectividad e interacción. Por ello, la base de Internet of Things son los sensores y las tarjetas RFID (Atzori et al., 2010; Tan, 2010). No obstante, no son los únicos. Pues según (Atzori et al., 2010), Near Field Communications (NFC), las

redes de sensores, Wireless Sensor and Actors Networks (WSAN) y RFID, “*son los componentes atómicos que unirán el mundo real con el mundo digital*”. Los sensores sirven para la recogida casi de cualquier tipo de información, cambio o dato. Este puede ser tratado por un Smart Object, como puede ser un Smartphone o enviado a un servidor. Por medio de las etiquetas inteligentes RFID se puede identificar diferentes objetos (Gama et al., 2012), para que, por medio de radiofrecuencia, se lean las propiedades de este elemento, e incluso, sea haga un seguimiento de él (Kortuem et al., 2010). Si combinamos ambos, se puede hacer que un objeto concreto realice una acción en base a un evento captado por un sensor determinado. Por ello se necesita de una infraestructura inteligente que sea capaz de conectar los diferentes objetos y les dote de la inteligencia necesaria para interactuar entre ellos, incluso, en algunos casos, indicándoles la decisión que deben tomar (Tan, 2010). Esto último, en el supuesto de que los Smart Objects no tengan inteligencia propia o necesiten en un momento dado una decisión externa.

Ejemplos aplicables del uso de Smart Objects en lo referente al ámbito comercial son (McFarlane, Sarma, Chirn, Wong, & Ashton, 2003) que proponen un sistema que ayude a controlar la facturación y (Meyer et al., 2009; Wong, 2002) uno para mejorar la distribución y gestión de productos teniéndolos localizados durante todo el proceso de su ciclo de vida. Ambos, muy útiles de cara a las empresas, otorgándoles ventajas para mejorar y evitar problemas de falta de stock a lo largo de toda la cadena de vida de un producto. Otro sistema es el de controlar el uso de objetos alquilados para cobrar según su utilización e incorrecto uso, propuesto por (Kortuem et al., 2010) y que ayuda tanto al cliente como a la empresa. En materia de seguridad, (Kortuem et al., 2010) propuso un sistema para avisar a los trabajadores acerca del almacenaje incorrecto e inseguro de materiales químicos. Este sistema puede ser muy útil, pues controlaría el almacenaje de posibles sustancias peligrosas y podría evitar muchos desastres y problemas. Para la salud, se propuso la monitorización de pacientes con problemas como se puede leer en (Akyildiz, 2002). Esto, aplicado correctamente, puede incluso llegar a salvar vidas humanas, al estar conectado con un centro de vigilancia y poder detectar inmediatamente el fallo de un marcapasos o un ataque al corazón.

## 5.2 *Smart Objects*

Como explican (Atzori et al., 2010; Hribernik et al., 2011; McFarlane et al., 2003; Wong, 2002) un Smart Object, también conocido como Intelligent Product, es un elemento físico, con diversas propiedades, identifiable a lo largo de su vida útil, que interactúa con el entorno y otros objetos y que puede actuar de manera inteligente según unas determinadas situaciones, mediante una conducta autónoma. Ejemplos de Smart Objects cotidianos son los Smartphones, las Tablets, Smart TVs e incluso algunos coches.

Los Smart Objects, según (Hribernik et al., 2011; Meyer et al., 2009) se pueden clasificar en base a tres dimensiones. Cada dimensión se corresponde a un tipo de inteligente. De esta manera, por medio de las tres dimensiones de un objeto, se puede determinar la inteligencia y tipo de Smart Object que es y compararlo con otros. Las dimensiones son las siguientes:

- La primera es el **nivel de inteligencia**. En esta se describe la capacidad de inteligencia del objeto. Consta de tres categorías: La **gestión de la información**: capacidad para manejar la información que recoge a través de sensores, lectores u otras técnicas; La **notificación del problema**, que es la posibilidad de que sea capaz de notificar a su propietario cuando ocurre un determinado problema o evento en el propio Smart Object, como puede ser la detección de bajada de temperatura; Por último, la **toma de decisiones**, que es cuando, además de todo lo anterior, posee la capacidad propia sobre la toma de decisiones sin intervención de un control externo.
- La segunda dimensión es la **localización de la inteligencia**. Esta consta de dos categorías. La primera es la **inteligencia a través de la red**. Consiste en que la inteligencia del objeto depende totalmente de un agente externo al propio objeto. Este agente puede ser una red a la que se encuentra conectado. La otra categoría es la **inteligencia en el objeto**. Los objetos que pertenecen a esta computan todo por sí mismo, es decir, toda la inteligencia es llevada en ellos.
- La última dimensión es la **agregación del nivel de inteligencia** y se compone de dos categorías: **inteligencia en el elemento** e **inteligencia en el contenedor**. A la primera pertenecen aquellos objetos que solo pueden manejar información, notificaciones y/o decisiones y si contienen otros componentes, estos no pueden ser distinguidos como objetos individuales. Por ejemplo, un sensor. Mientras, la inteligencia del contenedor, además de poder manejar información, notificaciones y/o decisiones, es capaz de seguir funcionando como elemento u objeto a pesar de que se le desensamble alguna parte de él. A este grupo pertenecería una placa Arduino con como mínimo tres sensores. Si a esta se le quita un sensor, puede seguir funcionando como contenedor.

De esta manera, se puede clasificar a un Smart Object en sus tres dimensiones. En esta investigación, se conectarán sensores con un nivel de inteligencia de gestión de la información y de notificación del problema, con la localización de la inteligencia a través de la red y de ambos niveles de agregación del nivel de inteligencia.

### 5.3 *Ingeniería Dirigida por Modelos*

La Ingeniería Dirigida por Modelos o MDE (Kent, 2002), surgió para solucionar los problemas pertenecientes al desarrollo software. Estos problemas son la baja calidad del software desarrollado, el incumplimiento del presupuesto y la planificación, y un aumento en el coste de mantenimiento del

proyecto. Problemas que ya surgían en los años 60 y que todavía se siguen dando, como bien se puede ver en (Dijkstra, 1972; González, Fernández, & Díaz, 2008). La solución a parte de estos problemas se puede obtener mediante la automatización o semi-automatización de procesos, algo en lo que es muy popular MDE (García-Díaz et al., 2010). Con ello se logra reducir la complejidad del diseño y de la implementación, lo que repercute en conseguir un software más fiable con funcionalidades más sofisticadas (Selic, 2008). Mediante el uso de MDE se puede proporcionar un aumento en la abstracción sobre los lenguajes de programación de tercera generación (C++, C#, Java,...). Esta abstracción se consigue mediante el uso de modelos. Esto otorga el uso de un concepto mucho más cercano al del dominio del problema al ser convertido a uno o más modelos los elementos del dominio (Núñez-Valdez, Sanjuan-Martinez, Bustelo, Lovelle, & Infante-Hernandez, 2013). Este modelo nos aporta una mayor facilidad para la creación de un Lenguaje de Dominio Específico (DSL). Por medio del uso de este DSL, se consigue aumentar la abstracción del problema, lo que repercute en un aumento de la productividad (Selic, 2008).

Para aplicar MDE se debe partir del dominio del problema que se quiere resolver y que delimita el campo de conocimientos. Estos dominios pueden ser subdivididos en unos dominios más pequeños. Por ello, el primer paso, es definir y acotar el dominio que se tratará. Por ejemplo: creación de aplicaciones que inserten datos en una base de datos, generación de videojuegos educativos de dos dimensiones en entornos móviles, etc...

Lo siguiente es definir el metamodelo. No obstante, primero se debe elegir el Meta-metamodelo que se utilizará. Lo que se consigue con el Meta-metamodelo es que el metamodelo creado sea reutilizable, interoperable y portable. Esto es debido a que es una instancia de una abstracción superior que define como debe ser el metamodelo. Los Meta-metamodelos se definen a sí mismo. Ejemplos de Meta-metamodelo son Ecore(Eclipse, 2013a) o Meta-Object Facility (MOF). Una vez elegido el Meta-metamodelo a utilizar, se define el metamodelo. En él hay que describir de manera formal los conceptos relevantes que tiene el dominio que se desea metamodelar.

Los metamodelos están formados de una sintaxis abstracta, una semántica estática y una sintaxis concreta. La sintaxis abstracta especifica su estructura, es decir, las construcciones, propiedades y conectores que puede poseer dicho lenguaje. En la Ilustración 1 se puede observar la sintaxis abstracta de Ecore. Esta posee en primer lugar las construcciones (paquetes, clases, tipos de datos, enumerables y anotaciones), después las propiedades (operaciones, atributos, literales y entradas) y al final los conectores (referencia, herencia y anotación).

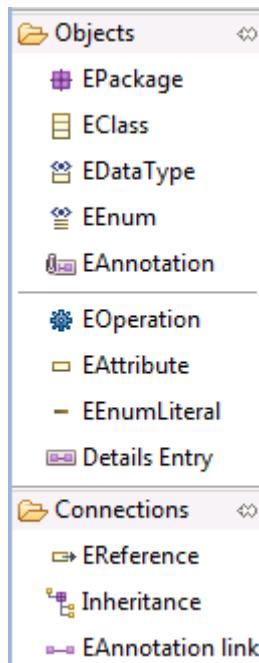


Ilustración 1 Sintaxis abstracta de Ecore

Una vez creado el metamodelo con la sintaxis abstracta, hay que realizar la transformación de esta a sintaxis concreta, de las cuáles puede haber más de una. Esta sintaxis concreta será la que utilicen los usuarios.

Para esta propuesta se utiliza la Ingeniería Dirigida por Modelos para crear un nivel de abstracción más alto y permitir la creación de un Lenguaje de Dominio Específico que facilite la generación de aplicaciones para interconectar objetos heterogéneos de una manera sencilla y rápida.

#### 5.4 Arquitectura Dirigida por Modelos

La arquitectura dirigida por modelos, conocida por sus siglas en inglés como MDA, surgió en el 8 de Marzo del año 2000 como iniciativa del grupo Object Manager Group (OMG) (Miller & Mukerji, 2003). El objetivo de la creación de MDA fue el de mejorar la productividad, la portabilidad, la interoperabilidad y la reutilización de los sistemas, así, como aumentar el nivel de abstracción. La arquitectura dirigida por modelos se basa, como su propio nombre indica, en el uso de modelos como forma para poder adaptarse a los cambios tecnológicos. Esta especificación define cuatro niveles de abstracción. La idea, es que, a partir del nivel de abstracción más alto, ir haciendo transformaciones automáticas o semi-automáticas hacia los niveles de menor abstracción. De esta forma, al final se llegará al código ejecutable por una máquina física o virtual. Las cuatro capas son:

- CIM (Computational-Independent Model): Esta primera capa se desarrolla sin preocuparse del sistema operativo o hardware final. En esta capa se muestran los detalles de la estructura

que se desarrollará sin tener en cuenta los aspectos informáticos. Esta capa se corresponde con el modelo del dominio de la aplicación.

- PIM (Platform-Independent Model): Esta capa se encuentra un nivel por debajo de CIM. EN la transformación de CIM a PIM, se crea un modelo pensando en un ordenador pero independiente de la plataforma. Ejemplos de esta capa son: Unified Modeling Language (UML), Meta-Object Facility (MOF) y Common Warehouse Metamodel (CWM).
- PSM (Platform-Specific Model): En este nivel se definen, a través de la transformación de PIM a PSM, las diferentes plataformas sobre las que se ejecutará el modelo inicial. Si en PIM el modelo era independiente de la plataforma, aquí se define cuáles son, por ejemplo: CORBA, XML, XMI, .NET, Java y Servicios Web.
- ISM (Implementation-Specific Model): Este es el nivel de abstracción más bajo. En él se define el código fuente específico del sistema para un entorno determinado creado a partir del PSM correspondiente. Algunos ejemplos de esta capa sería, referentes a los PSM: CORBA (Iona Orbix, Borland VisiBroker) y en Java (BEA WebLogic Server, IBM WebSphere software platform).

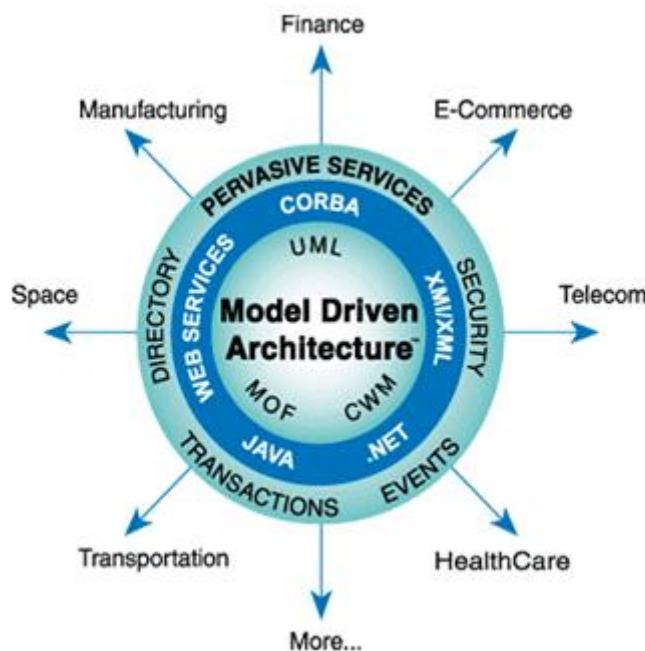


Ilustración 2 Ejemplo de MDA distribuido por capas según el OMG

En esta investigación, debido a la innecesidad de utilizar todo el estándar MDA, se utilizó MDE. Esto es debido a que no eran necesarias todas las transformaciones internas automáticas o semi-automáticas entre capas que propone MDA. En nuestro caso se buscaba crear un prototipo que cumpliese MDE para poder obtener la abstracción del problema de cara al usuario pero sin la necesidad de introducir un

proceso de automatización en todas las capas, partiendo del metamodelo. Esta automatización, en nuestro caso, se necesita sólo a partir del editor web gráfico, que se correspondería con la capa PSM. No obstante, es cierto que para MDA hay herramientas que facilitan mucho el desarrollo de estas capas, pero como se explicó, al no requerir el uso de todo el estándar, se consideró que era mucho más óptimo crear una herramienta propia que cumpliese MDE que utilizar un framework para trabajar con MDA.

### 5.5 *Lenguajes de Dominio Específico*

Un Lenguaje de Dominio Específico o DSL es un lenguaje, normalmente declarativo y que hace llamadas a sub-procesos, que permite solucionar un determinado problema específico en un dominio concreto. Así, los DSL se caracterizan por tener un gran poder expresivo (Arie Van Deursen et al., 2000; Núñez-Valdez et al., 2013). Las ventajas que ofrece el uso de un DSL son la gran mejora en la productividad, la confiabilidad de cometer menos errores, la facilidad de su mantenimiento (Reed, Halpern, & Starr, 1996), portabilidad, conocimiento sobre el dominio del problema y la reutilización para diferentes propósitos (A Van Deursen, 1997; Arie Van Deursen et al., 2000). Como contras tiene la disminución de la eficiencia frente a la codificación nativa y la dificultad para crear el dominio y construir el DSL (Arie Van Deursen et al., 2000; Reed et al., 1996). No obstante, los Lenguajes de Dominio Específico son muy importantes en la Ingeniería Dirigida por Modelos (Núñez-Valdez et al., 2013). Algunos ejemplos de Lenguajes de Dominio Específico ampliamente utilizados son: Unix Shell Script, ColdFusion Markup Language, MediaWiki Templates, Unreal Engine, AutoLisp, AutoCAD, HTML, SGML, Mathematica, SQL, LATEX, MOF, EBNF, CSS,....

Para esta propuesta se aporta un Lenguaje de Dominio Específico que permita la creación de aplicaciones para interconectar objetos heterogéneos por medio de una abstracción sobre la codificación nativa de estas. Esto, junto a un editor web gráfico, proporcionará la posibilidad de crear dichas aplicaciones a usuarios inexpertos en el dominio de la programación de aplicaciones informáticas pero que tengan conocimiento del dominio de los objetos heterogéneos.

### 5.6 *eXtensible Markup Language*

eXtensible Markup Language, conocido por sus siglas, XML (World Wide Web Consortium, 2004), es un lenguaje de marcas. Este estándar fue creado por el W3C para almacenar datos de forma legible para permitir el intercambio de información estructurada entre diferentes plataformas. Una de las grandes ventajas que presenta XML es la posibilidad de extender el lenguaje por medio de la creación de etiquetas. Además, cuenta con muchas tecnologías que lo complementan, otorgándole unas posibilidades mucho mayores.

### 5.7 *XML Metadata Interchange*

XML Metadata Interchange, mejor conocido como XMI (Object Management Group, 2005), es una especificación basada en XML creada por el OMG para compartir metadatos entre herramientas de modelado UML. XMI integra cuatro estándares. Estos son XML, UML, MOF y la correspondencia entre MOF y XMI. La gran ventaja del uso de XMI es que se puede almacenar cualquier modelo basado en MOF. No obstante, XMI es un formato difícil de leer por los seres humanos, lo que dificulta su uso por estos, haciendo de esto una gran desventaja.

### 5.8 *XML Schema*

XML Schema (XSD), es un lenguaje utilizado para describir tanto la estructura como las restricciones del contenido de un documento XML. XSD se basa en el espacio de nombres. Estos contienen elementos y atributos que están relacionados con este espacio de nombres. De esta manera, cuando se define un elemento o atributo se crea una conexión entre los diferentes campos de este. Tras definir un Schema XML, este se puede validar para comprobar la correcta definición de este.

XSD ofrece un nivel de detalle alto y la posibilidad de validarla. No obstante, esto implica una mayor complejidad a la hora de definirla. En el caso de MIDGAR, todo se trata como una cadena de caracteres debido a la capa de abstracción dotada a los objetos para que puedan interconectarse sin importar el tipo de dato. Esto hace que la definición exacta de los tipos de los diferentes campos, en el caso de MIDGAR, sea indiferente, pues, estos se tratan todos en el servidor según lo especificado en el registro de cada servicio, siendo totalmente indiferente a la hora de crear la aplicación.

En el caso de MIDGAR, todo se trata como una cadena de caracteres debido a la capa de abstracción creada en los objetos para que puedan interconectarse sin importar el tipo de dato. Esto hace que la definición exacta de los tipos de los diferentes campos, sea indiferente, pues, estos se tratan todos en el servidor según lo especificado en el registro de cada servicio, siendo totalmente indiferente a la hora de crear la aplicación.

### 5.9 *XSLT*

Extensible Stylesheet Language Transformations (XSLT) es un estándar desarrollado por el W3C (World Wide Web Consortium, 1999). XSLT es el estándar que muestra una forma de transformar documentos XML en otros, e incluso, en otros formatos (PDF, Latex, HTML, XHTML,...). Para ello, se usan varias reglas de una plantilla a través de un procesador XSLT que se encarga de realizar las transformaciones. Si se utiliza en conjunto con XML, permite separar el contenido (XML) de la presentación (XSLT).

Al utilizar XML para almacenar la información de la aplicación desarrollada por el usuario, se podría haber utilizado una plantilla XSLT para la transformación del XML en la aplicación. No obstante, se prefirió utilizar una aplicación creada con el lenguaje Java debido a los conocimientos previos adquiridos en anteriores proyectos con el generado y procesado de XML.

## 5.10 *Plataformas IoT*

Actualmente existen varias plataformas para interconectar dispositivos. Entre ellas se encuentran algunas orientadas al mundo empresarial, como es el proveedor de servicios Xively. Otras están orientadas a la investigación, como son Paraímpu y QuadraSpace. Sobre Paraímpu se puede encontrar varias publicaciones en diferentes sitios. Estas tratan los problemas principales de IoT. Por último, están las redes abiertas a los usuarios pero que están en estado beta y limitan el acceso a los usuarios mediante invitación o aprobación de la cuenta. Ejemplos de esto son ThingSpeak, Sensorpedia o SenseWeb. En esta sección se describirán Paraímpu, Xively y ThingSpeak. La elección es clara: son las que ofrecen más datos para realizar la investigación de sobre cómo funcionan y las que se pudieron probar. Las demás redes comentadas no fueron accesibles debido a su estado beta o no proporcionaban información suficiente. Las tres redes estudiadas comparten algo en común: permiten generar disparadores a los usuarios para interconectar dos objetos en base a una condición. Disparadores sencillos y limitados: una condición, una acción.

### 5.10.1 *Paraímpu*



Ilustración 3 Logotipo de Paraímpu

Es una plataforma capaz de integrar diversos Smart Objects en una misma red capaz de administrar los datos heterogéneos, compartirlos y conectar diferentes sensores.

Esta plataforma permite a los usuarios conectarse, compartir y componer aplicaciones para conectar objetos.

Permite trabajar con datos en formatos XML, JSON, cadenas de caracteres y datos numéricos a través del protocolo HTTP. Divide los objetos en dos tipos: sensores y actuadores. Los primeros son aquellos objetos capaces de recoger datos y enviarlos a la plataforma y los actuadores son los objetos que pueden realizar una acción. Para ello proporciona a los usuarios clases predefinidas para utilizar 10 sensores y 13 actuadores. Entre ellas se encuentran las clases para utilizar Arduino, Twitter, Pachube y Foursquare. Puede establecer conexión directa entre dos objetos heterogéneos siempre que estos incorporen el mecanismo de paso de mensajes de la plataforma. Para generar las aplicaciones, el usuario establecer la condición que quiere que se cumpla indicando el nombre del sensor, su valor y la condición y después el resultado a ejecutarse: “*Match: NombreDelSensor.value > 25*” y “*Replace: ‘today is <%new Date().toString()%> and we have <% NombreDelSensor.value%> <% NombreDelSensor*

.unit%> ‘’. Permite configuración sobre el nivel de privacidad a la hora de compartir los sensores (Pintus et al., 2011; Pintus, Carboni, & Piras, 2012a, 2012b; Piras et al., 2012).

Además, como se puede observar en el ejemplo que ellos proporcionan, el usuario debe conocer ciertas reglas de programación y utilizar un lenguaje bastante orientado a la programación, como es la creación de un objeto (new), la introducción de script (<%%>) o el acceso a una propiedad o un método (.). Consiguieron reducir mucho la complejidad de programar una aplicación pero el usuario todavía debe utilizar métodos cercanos a la programación. Cabe destacar que solo permiten la creación de condiciones, reduciendo también otras muchas posibilidades que puede ofrecer otras estructuras de control para manejar los objetos.

### 5.10.2 Xively



Ilustración 4 Logotipo de

Anteriormente conocida como COSM y previamente como Pachube. Xively es un proveedor de servicios en la nube. Esta red IoT proporciona la posibilidad de subir datos de diferentes Smart Objects u objetos utilizando las librerías que ellos proporciona. Para utilizarlas se debe utilizar la API Key que te asignan cuando te registras o crear una nueva. Con ella se puede limitar el uso de ciertas acciones REST a un dispositivo.

En total dan diez librerías, entre las que cabe destacar la de Arduino, Java y Android. Por medio de su uso, un usuario puede subir a su perfil los datos de diferentes objetos. En su perfil, un usuario puede añadir dispositivos. A este nuevo dispositivo, puede añadirle disparadores para que, tras cumplirse una determinada condición realice una petición post a un servicio HTTP usando un método REST para comunicarle la acción (LogMeIn, 2013). En la Ilustración 5 se puede ver la creación de un disparador en esta página.

A screenshot of the Xively 'Triggers' interface. It shows a modal window titled 'Add Trigger'. The 'Channel' dropdown is set to 'channel'. The 'Condition' section shows a comparison operator 'greater than' (>) and a value '25'. The 'HTTP POST URL' field contains 'e.g. http://example.com'. At the bottom are 'Save Trigger' and 'Cancel' buttons.

Ilustración 5 Creación de un disparador en Xively

Estado del Arte

Xively proporciona grandes posibilidades de personalización de la privacidad y acceso mediante REST a otro servicio, así como facilita mucho la conexión de objetos mediante su creador de disparadores. Además estos, están bastante enfocados a usuarios no conocedores de los conceptos de la programación. No obstante, solo permite la realización de una sola condición y reduce bastante las posibilidades.

### 5.10.3 ThingSpeak

ThingSpeak es una red IoT para conectar objetos. Esta ofrece tutoriales sobre como conectar diferentes objetos y servicios. Lo primero que hay que hacer es crear un nuevo dispositivo seleccionando el tipo (Arduino, Netduino, ioBridge,...) y poniendo los datos de acceso de este (ip, puerto, submáscara,...). Tras esto permite crear un nuevo canal. Los canales sirven para subir los datos de los dispositivos conectados, mostrarlos gráficamente y acceder a sus datos mediante una petición REST para bajarte los datos en formato XML (Ilustración 7), CSV (Ilustración 8) o JSON (Ilustración 9). Para poder interactuar en base al objeto deseado, hay que bajar los datos en uno de estos tres formatos y procesarlo para así obtener los datos necesarios.



Ilustración 6 Logotipo de ThingSpeak

```
<channel>
  <name>my_house</name>
  <created-at type="datetime">2010-12-14T01:20:06Z</created-at>
  <updated-at type="datetime">2013-07-04T15:07:35Z</updated-at>
  <id type="integer">9</id>
  <last-entry-id type="integer">4777811</last-entry-id>
  <description>
    Netduino Plus connected to sensors around the house
  </description>
  <latitude type="decimal">40.44</latitude>
  <longitude type="decimal">-79.996</longitude>
  <field1>Light</field1>
  <field2>Outside Temperature</field2>
  <feeds type="array">
    <feed>
      <created-at type="datetime">2013-07-03T15:07:52Z</created-at>
      <entry-id type="integer">4772211</entry-id>
      <field1>475</field1>
    </feed>
```

Ilustración 7 ThingSpeak: Datos en formato XML

created_at,entry_id,field1
2013-07-03 15:09:52 UTC,4772219,474
2013-07-03 15:10:07 UTC,4772220,472
2013-07-03 15:10:22 UTC,4772221,480
2013-07-03 15:10:37 UTC,4772222,469
2013-07-03 15:10:52 UTC,4772223,471
2013-07-03 15:11:07 UTC,4772224,460
2013-07-03 15:11:22 UTC,4772225,478
2013-07-03 15:11:37 UTC,4772226,478
2013-07-03 15:11:52 UTC,4772227,480
2013-07-03 15:12:07 UTC,4772228,469

Ilustración 8 ThingSpeak: Datos en formato CSV

## Estado del Arte

```
{"channel": {"created_at": "2010-12-14T01:20:06Z", "description": "Netduino Plus connected to sensors around the house", "field1": "Light", "field2": "Outside Temperature", "id": 9, "last_entry_id": 4777814, "latitude": "40.44", "longitude": "-79.996", "name": "my_house", "updated_at": "2013-07-04T15:08:20Z"}, "feeds": [{"created_at": "2013-07-03T15:08:37Z", "entry_id": 4772214, "field1": "485"}, {"created_at": "2013-07-03T15:08:52Z", "entry_id": 4772215, "field1": "462"}, {"created_at": "2013-07-03T15:09:07Z", "entry_id": 4772216, "field1": "469"}, {"created_at": "2013-07-03T15:09:22Z", "entry_id": 4772217, "field1": "465"}, {"created_at": "2013-07-03T15:09:37Z", "entry_id": 4772218, "field1": "477"}], {"created_at": "2013-07-03T15:09:52Z", "entry_id": 4772219, "field1": "474"}]
```

Ilustración 9 ThingSpeak: Datos en formato JSON

Por el contrario, para enviar datos hay que crear una petición mediante protocolo HTTP, método POST y siguiendo la arquitectura REST:

```
http://api.thingspeak.com/update?key=(API Key del canal de ThingSpeak donde se desea escribir)&field1=(Dato 1)&field2=(Dato 2)&field3=(Dato 3)&field4=(Dato 4)&field5=Dato 5)&field6=(Dato 6)&field7=( Dato 7)&field8=(Dato 8)&lat=(Latitud en grados decimales)&long=(Longitud en grados decimales)&elevation=(Elevación en metros)&status=(140 caracteres como máximo de mensaje)
```

Una vez los datos están subidos, se puede elegir la opción de que la plataforma cree gráficas interactivas para la visualización de los datos (Ilustración 10). Ofrece también un sistema de privacidad para los objetos para poder establecer si son accesibles por todo el mundo o son privados para el usuario (IoBridge, 2013).

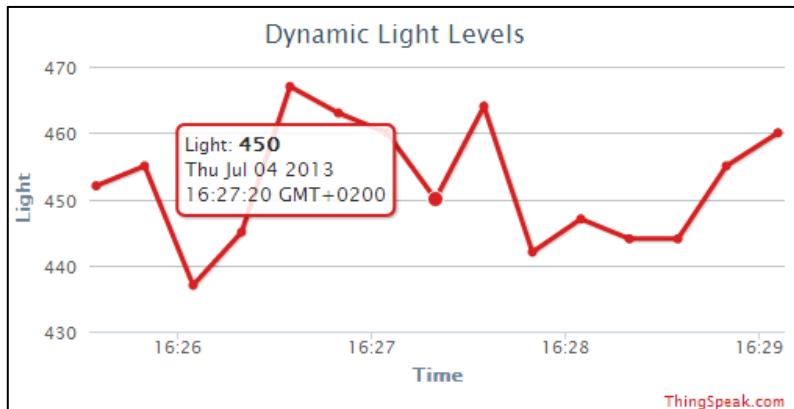


Ilustración 10 Gráfica generada por ThingSpeak con información sobre la luz

Esta plataforma proporciona un buen sistema de visualización de los datos proporcionando gráficas interactivas. Por ejemplo, en la Ilustración 10 se puede ver como representa cada llegada de un dato con un punto y los conecta mediante una línea recta. Después cada punto contiene la información respecto al dato enviado y la fecha en que se realizó, la cual se muestra de manera interactiva cuando se pone el cursor del ratón sobre él. No obstante, para conectar diferentes dispositivos es necesario bajar los datos en bruto de estos y procesarlos (a pesar de la existencia de librerías que facilitan el trabajo). Esto requiere de bastante trabajo y de conocimientos de programación, pues hay que realizar la aplicación.

## 6 CASO DE ESTUDIO: MIDGAR

MIDGAR es una plataforma de Internet of Things desarrollada específicamente de cara a la investigación de los problemas de estas plataformas y de la interconexión e interoperabilidad de objetos. En este trabajo final de máster se intenta dar una solución a la generación de aplicaciones que interconecten objetos heterogéneos. En este apartado se describirá la plataforma MIDGAR y la solución adoptada para los problemas comentados.

### 6.1 *Generación de aplicaciones interconectores de objetos heterogeneos*

Con el Lenguaje de Dominio Específico (DSL) que se presenta, el usuario podrá interconectar diferentes objetos, de una manera sencilla y sin requerir conocimientos de programación. Para poder interconectarlos, el usuario puede utilizar el editor gráfico realizado para escribir la aplicación utilizando el DSL creado para este caso. Si así lo desea, puede no utilizar el editor otorgado. No obstante, como se comentará en la

Evaluación y discusión, este le ayudará de una manera bastante óptima en la creación del DSL. Gracias a este DSL, un usuario puede describir todo el flujo que desee que realice la aplicación. En ella es posible conectar multitud de objetos y leer de él cualquier parámetro de un servicio que estos posean y del que suban datos. Siempre que estos servicios se encuentren registrados en la plataforma. De esta manera el usuario puede leer datos de varios objetos y según estos, mandar realizar ciertas acciones a los mismos u otros diferentes. Tantas acciones y condiciones como el usuario deseé.

Dichas acciones pueden diferir entre unos objetos y otros. Estas se registran al mismo tiempo que el objeto y los servicios. Un objeto puede tener infinidad de acciones, al igual que servicios. Así como una aplicación que interconecte objetos puede tener infinidad de objetos conectados y de acciones a realizar.

## 6.2 *Arquitectura propuesta*

La arquitectura del sistema se puede dividir en cuatro capas como se ilustra en la Ilustración 11Figure 1;**Error! No se encuentra el origen de la referencia..** Cada una es un proceso dentro del conjunto global dentro de la infraestructura. Así, la primera capa, abarca el proceso del usuario. En este, un usuario cualquiera mediante el uso del editor web debe definir lo que quiere que haga su aplicación. Tras terminarla, se genera en formato XML el DSL que contiene la información de esta. La información es pasada a la segunda capa: la capa de generación del servicio. Esta capa se encarga de procesar la información, generar una aplicación a partir de ella, compilarla y ejecutarla en el servidor. Este último paso forma parte de la capa tres, la capa del servidor. De esta manera, la aplicación creada por el usuario queda funcionando en el servidor realizando lo que el usuario haya definido. Mientras está en funcionamiento, se comunica directamente con el servidor y la base de datos y establece, si se debe, los mensajes a enviar al objeto requerido la próxima vez que este se conecte. Esta capa además contiene también el servicio al que se deben de conectar los Smart Objects que interactúan con MIDGAR. Estos objetos deben implementar un paso de mensajes concreto para mantener la conexión con el servidor. Por medio del procesado de estos mensajes, el servidor obtiene los datos de cada objeto. Esto permite a los servicios creados por los usuarios interactuar con el servidor y dejarle los mensajes para los objetos. En la última capa se encuentran los Smart Objects. Estos, implementan la interfaz de mensajes para así poder mantener una conexión permanente y bidireccional con el servidor.

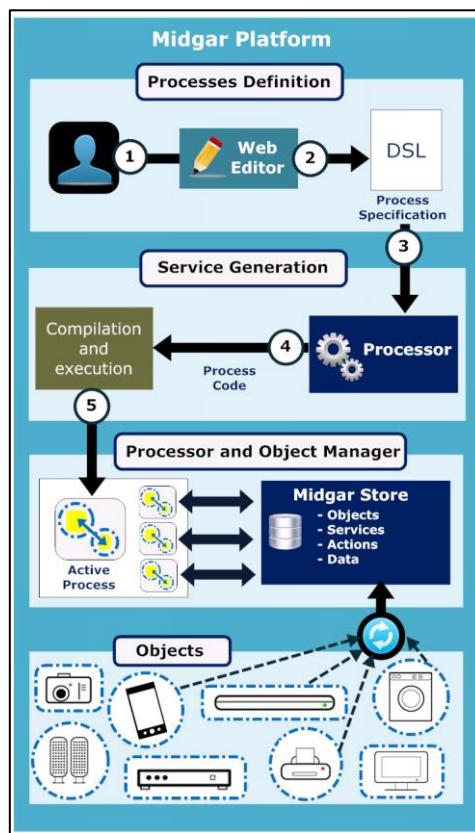


Ilustración 11 Arquitectura de la plataforma MIDGAR

### 6.3 *Modelo obtenido aplicando Ingeniería Dirigida por Modelos*

Para crear el Lenguaje de Dominio Específico, primero había que crear el modelo del problema perteneciente a nuestro dominio: Creación de aplicaciones por usuarios no expertos para interconectar dispositivos en una red IoT.

Como Meta-metamodelo se eligió Ecore (Eclipse, 2013a) debido a la facilidad de uso y creación que ofrece, así como las posibilidades que otorga el proyecto Eclipse Modeling Framework (Eclipse, 2013b) con las Ecore Tools (Eclipse, 2013c) y el previo conocimiento de su uso en otras ocasiones y utilización en la propuesta (González García & Pascual Espada, 2013).

El siguiente paso fue definir el Metamodelo a partir del Meta-metamodelo. Este se puede ver en la Ilustración 12. En este Metamodelo se define de manera formal los conceptos relevantes que tiene nuestro dominio. Para crearlo, se pensó en que necesitaría el usuario para poder generar una aplicación que contuviese todo lo que otorga un lenguaje de programación.

Caso de estudio: MIDGAR

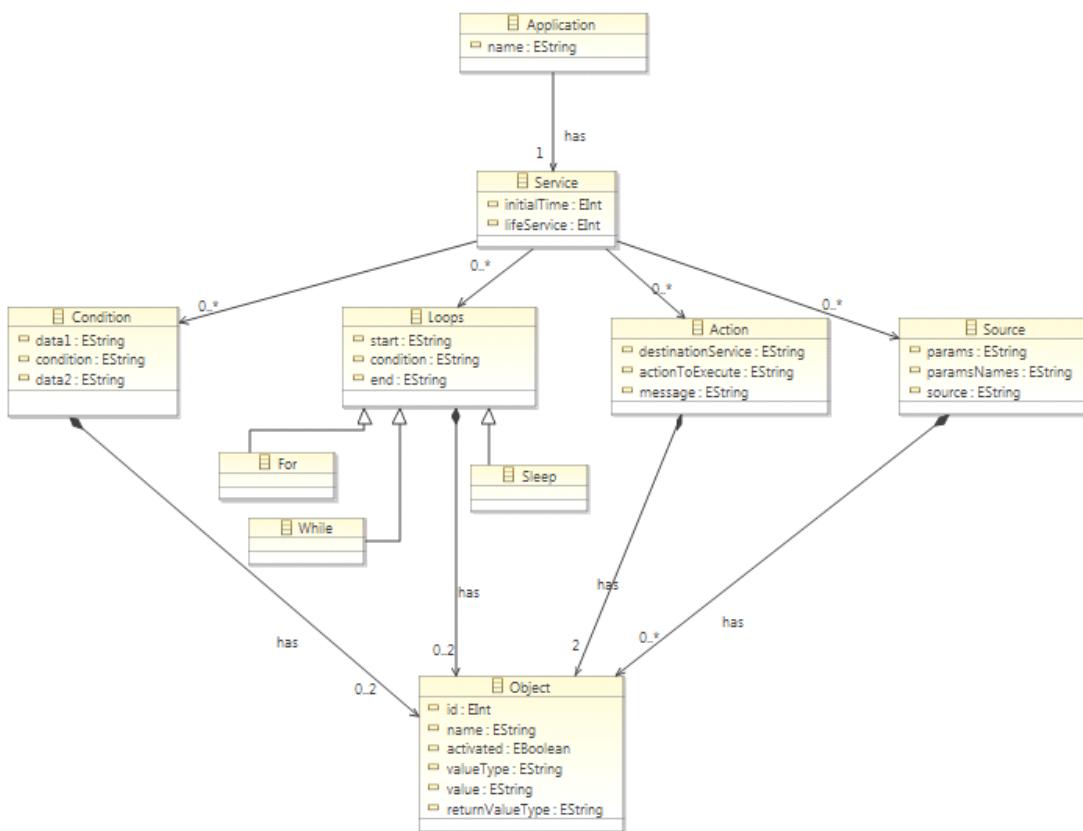


Ilustración 12 Metamodo

Las diferencias respecto a la propuesta realizada inicialmente, son bastantes, aunque no relevantes. Primero, se optó por eliminar al usuario y dejar, lo que se conoce como Internet of Things Social (SIoT) (Atzori et al., 2012) para el trabajo futuro debido a la gran cantidad de trabajo e investigación que esta parte requeriría. El “*Component*” pasó a convertirse en “*Object*”. Este cambio viene fundamentado en que se consiguió realizar un prototipo de modelo previo para tratar a todos los objetos por igual (Véase apartado 6.4.6), indiferentemente del tipo de objeto (Smartphone, micro-controlador). No obstante, este modelo no se presenta debido a que es un prototipo muy básico y la investigación de este tema se prefirió dejar para el trabajo futuro debido a la cantidad de investigación que este requeriría. La “*Action*” perdió la condición debido a que ahora el usuario no define una sola condición como se hace en otras plataformas (Véase apartado 5.10). Ahora el usuario puede definir todo el flujo que quiere que realice la aplicación, puede interconectar infinidad de objetos e incluso realizar diferentes tipos de bucles (iteración creciente, decreciente, de pausa) e incluso, insertar en ellos valores de otros objetos. En resumen, este cambio viene debido a que ahora se ofrece una mayor personalización y libertad al usuario y no se le limita a una simple condición. Sobre las “*Networks*”, desapareció en favor de dejar una única red y dejar ese tema para cuando se investigue SiOT.

Una vez realizado el metamodelo, se hizo la conversión de la sintaxis abstracta a la sintaxis concreta. Primeramente, a la sintaxis concreta del DSL Textual. Más adelante, cuando se fue a realizar el editor gráfico, se realizó primero una conversión a otra sintaxis concreta para el DSL gráfico. Las equivalencias se pueden ver en la Tabla 1. Cabe destacar que, en el DSL Gráfico, los nodos “*application*” y “*service*” se fusionaron en uno para facilitar la generación de aplicaciones.

Sintaxis abstracta	Sintaxis concreta	
	DSL Textual	DSL Gráfico
<b>Application</b>	<application>	
<b>Service</b>	<service>	
<b>Condition</b>	<ifCondition>	
<b>For</b>	<forLoop>	
<b>While</b>	<whileLoop>	
<b>Sleep</b>	<sleep>	
<b>Action</b>	<action>	
<b>Source</b>	<java>	

Tabla 1 Correspondencia entre la Sintaxis Abstracta y la Sintaxis Concreta

#### 6.4 *Implementación*

En este sub-apartado se describirá en detalle los diferentes componentes y capas de la plataforma y cómo interactúan entre ellas. Se comenzará describiendo el editor web explicando cómo funciona. Tras esto, e pasará a explicar en detalle el Lenguaje de Dominio Específico creado. Ambos componentes pertenecientes a la primera capa y que la enlazan con la segunda. Esta será la siguiente. En ella se explicará cómo se procesa el DSL y como genera la aplicación residente en el servidor. Se continuará

#### Caso de estudio: MIDGAR

hablando de la capa Processor and Object Manager y sus dos componentes: El servidor de procesos y el almacén de datos. Para terminar, se explicará cómo funciona la conexión de los objetos con la plataforma.

##### 6.4.1 Editor Web Textual

La primera versión del editor realizada, fue un editor de texto (Ilustración 13). En esta primera versión se ofrecía al usuario ayuda para crear el Lenguaje de Dominio Específico. Los nodos “*application*” y “*service*” ya vienen creados por defecto. Así, el usuario lo único que deberá hacer es completar estos e insertar dentro de ellos el contenido que quiera para crear la aplicación. Para esto, deberá colocar el cursor del teclado en la zona donde desee insertar texto y seleccionar una de las opciones de la izquierda. Esto hará que ese nodo se le inserte donde tenía situado el cursor. Con esto se consigue que el usuario sólo deba llenar los atributos. El editor se puede dividir en tres zonas.

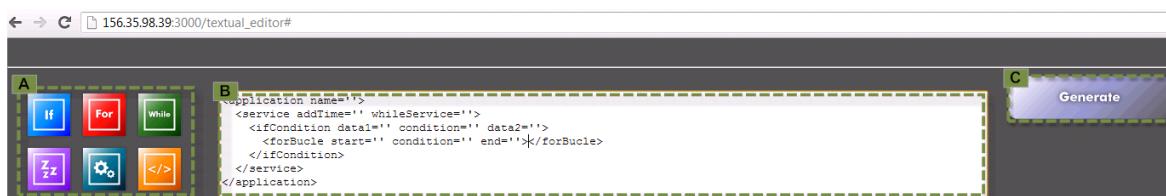


Ilustración 13 Editor textual para la generación de aplicaciones interconectoras de objetos de la plataforma MIDGAR

La primera zona (Ilustración 14Figure 2Ilustración 13A) contiene el panel de selección con los botones para incluir un elemento en el flujo de la aplicación. De izquierda a derecha y de arriba abajo son: El “*if*”, incluye una condición al flujo; el “*for*”, que permite la generación de bucles que se ejecuten mientras se cumple una condición; el “*while*”, otro bucle muy parecido al “*for*” pero que da soporte a las condiciones basadas en tiempo; el “*sleep*” que permite hacer que la aplicación creada duerma durante un tiempo determinado; la acción, el elemento final y más importante, pues, mediante su inclusión, se permite enviar una orden junto a un mensaje a un objeto para que la execute; inserción de código, en este caso se permite la opción de incluir código en la aplicación que permite incluir cualquier tipo de flujo soportado por el lenguaje base (Java) y realizar peticiones de interacción con objetos. Como ejemplo se puede ver la Ilustración 13B. La aplicación aquí creada tiene los nodos “*application*”, “*service*”, “*if*” y “*for*”. Fueron insertados como se explicó antes, pero contienen los atributos sin llenar. Gracias a este editor, esto es lo único que el usuario debe hacer.

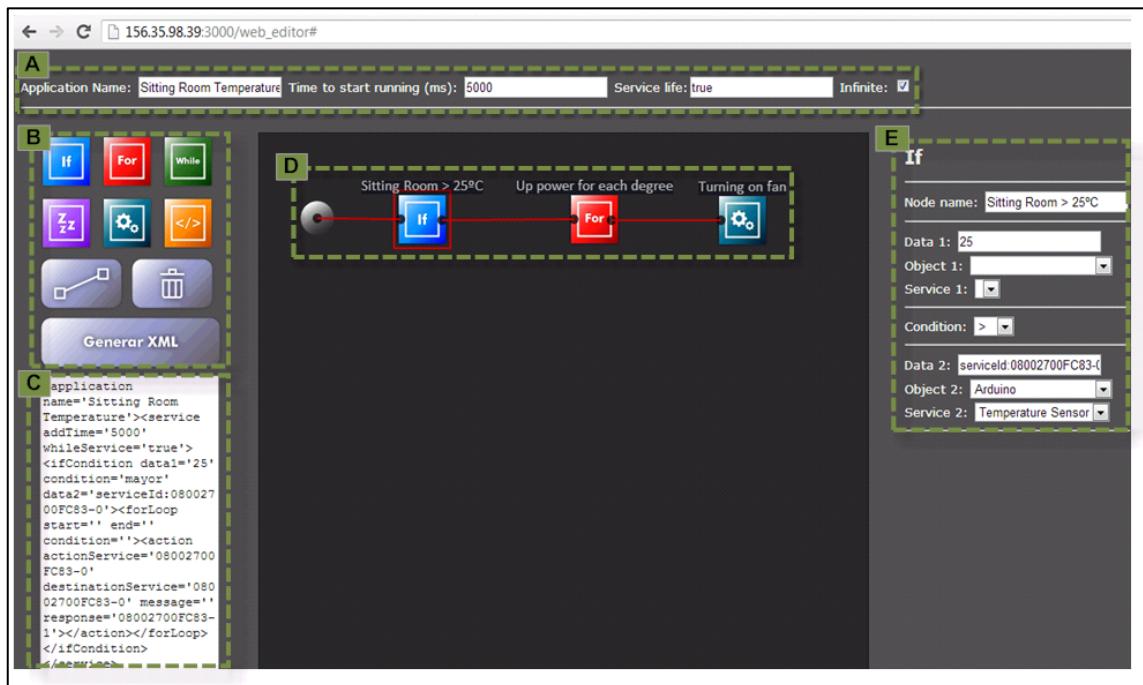
Por último, en la Ilustración 13C se ve la zona de los formularios. Por último, el botón generar permite la generación de esta aplicación.

##### 6.4.2 Editor Web Gráfico

El editor final creado para el proyecto MIDGAR es un editor gráfico (Ilustración 14;**Error! No se encuentra el origen de la referencia.**). De esta forma, si el usuario desea generar una aplicación, lo que

Caso de estudio: MIDGAR

debe hacer es seleccionar la caja correspondiente al flujo que desea añadir al programa y enlazarla al resto el uso de los conectores. El editor se puede dividir en cinco zonas.



**Ilustración 14 Editor gráfico para la generación de aplicaciones interconectoras de objetos de la plataforma MIDGAR**

La primera se corresponde a la Ilustración 14Figure 2A. Esta zona contiene los datos correspondientes a la aplicación que se desea crear: nombre, tiempo de espera hasta comenzar a ejecutarse desde el momento en que se crea y el tiempo de vida de la aplicación. En este último se puede indicar mediante el checkbox su ejecución infinita.

La segunda zona (Ilustración 14Figure 2B) contiene el panel de selección con los botones para incluir un elemento en el flujo de la aplicación. De izquierda a derecha y de arriba abajo son: El “if”, incluye una condición al flujo; el “for”, que permite la generación de bucles que se ejecuten mientras se cumple una condición; el “while”, otro bucle muy parecido al “for” pero que da soporte a las condiciones basadas en tiempo; el “sleep” que permite hacer que la aplicación creada duerma durante un tiempo determinado; la acción, el elemento final y más importante, pues, mediante su inclusión, se permite enviar una orden junto a un mensaje a un objeto para que la ejecute; inserción de código, en este caso se permite la opción de incluir código en la aplicación que permite incluir cualquier tipo de flujo soportado por el lenguaje base (Java) y realizar peticiones de interacción con objetos. El siguiente botón es el de conexión de nodos. Mediante el uso de este botón se crean las conexiones entre los elementos incluidos en la aplicación y se establece así el flujo y orden que debe seguir la aplicación desarrollada. A su derecha está el botón de

eliminar. Con él se puede eliminar cualquier elemento del flujo. Por último, el botón generar permite la generación de esta aplicación y la visualización del DSL en el editor.

En la Ilustración 14Figure 2C se encuentra la visualización del DSL correspondiente a la aplicación creada por el usuario con el editor. En la zona central se encuentra la zona de trabajo. Aquí es donde se crea la aplicación y se conecta el flujo que se desea realizar. El flujo de la aplicación siempre debe empezar a partir de la bola gris. Como ejemplo se puede ver la Ilustración 14Figure 2D. La aplicación aquí creada comprueba el valor de temperatura del micro-controlador Arduino. Si este es mayor de 25°C crea un bucle en el que, por cada grado por encima de 25, aumenta la velocidad del motor conectado al Arduino y que maneja un ventilador encontrado en esa habitación.

Por último, en la Ilustración 14Figure 2E se ve la zona de los formularios. En ella se visualiza el formulario correspondiente al nodo seleccionado en la zona de trabajo. En este caso esta seleccionado el nodo if.

#### **6.4.3 Lenguaje de Dominio Específico**

El Lenguaje de Dominio Específico creado para la generación de aplicaciones que interconecten dispositivos cuenta con un total de seis nodos. Cuatro de ellos se corresponden con estructuras del lenguaje de programación, uno con la inserción de código directo del propio lenguaje y otro con la acción a ejecutarse. Los cuatro primeros son el condicional, dos bucles y el “sleep”. Por medio de estos, el usuario puede crear el flujo de ejecución del programa ya sea, insertando condicionales para que solo se ejecute bajo una o varias condiciones, un bucle para que se ejecuten una o más tareas varias veces o controlar el flujo de tiempo de ejecución de programa insertando pausas o esperas. Para crear el flujo del programa pro medio del uso del DSL, el usuario debe ir anidándolos unos dentro de otros si lo quede sea es que estos se ejecuten cuando se cumpla el padre, en caso de ser el condicional o ejecute eso si el padre es un bucle. Si lo que desea es que se ejecute uno detrás de otro, simplemente debe colocarlos seguidos.

El DSL tiene que tener siempre un encabezado predefinido que contiene el nombre de la aplicación, el tiempo, en milisegundos, de espera para que se inicie la aplicación, contado desde cuando se crea. El otro parámetro indica si esta debe ejecutarse por siempre o un solo por un tiempo determinado.

Caso de estudio: MIDGAR

---

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

El siguiente código se corresponde con una de las primeras aplicaciones realizadas. El dato cogido del primer servicio se correspondía con el eje Y del acelerómetro del móvil. Cuando este se ponía casi vertical y superaba dicho valor, se enviaba un mensaje a otro móvil. En este primer ejemplo, se ejecutaba una acción siempre y cuando se cumplía la condición. Si el resultado devuelto por el servicio dos del dispositivo *e6644a22aae2cec6* es mayor que ocho, el servicio cero del dispositivo *c3b9f28c24f2be8b* ejecuta la acción (actionToExecute) dos y se le envía el mensaje “*This action is a popup*”. En este caso, lo que hace es ejecutar la acción dos del dispositivo, que es un mensaje por pantalla que muestra el mensaje recibido.

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

Otro ejemplo es el de usar un bucle para enviar varias acciones seguidas pero con una pausa entre cada acción de varios segundos. El siguiente ejemplo, se corresponde con una aplicación realizada que leía la temperatura de un sensor que se encontraba colocado en un micro-controlador Arduino. Cuando este superaba los 25°C, enviaba por cada grado de diferencia una vibración de 2500 milisegundos al móvil personal del usuario. Este servicio puede resultar útil para controlar sitios en los que pueda subir mucho la temperatura y avise de una forma un poco intrusiva al controlador de la zona. Otro ejemplo de uso podría ser la cocina y, que en vez de enviarlo a un móvil, lo enviase a otro micro-controlador Arduino que dispusiese de un altavoz y así poder avisar mediante una alarma sonora al cocinero. Como se ve en el código, cuando el valor del servicio dos del objeto *08002700FC83* supera el valor 25 se ejecuta un bucle que va desde veinticinco hasta el número devuelto mismo servicio. Una vez entra, ejecuta la acción desde cero con un mensaje de 2500. En este caso esta acción es la vibración con una duración de 2500 milisegundos. Tras ejecutar esta acción, ejecuta una similar a la del ejemplo anterior, un mensaje y después espera cinco segundos para enviarle de nuevo la misma acción al dispositivo.

```
<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>
```

Como se pudo observar en el ejemplo anterior, las dos acciones y el “*sleep*” se ejecutan dentro del cuerpo del bucle, al igual que este se ejecuta sólo cuando se cumple la condición. Por eso van anidados. No obstante, las acciones y el “*sleep*” se ejecutan una detrás de otra pues no están anidadas entre ellas.

El usuario también puede realizar un bucle “*while*” e incluso introducir código Java directamente en la aplicación. En el segundo caso, esta etiqueta viene bien si el usuario posee conocimientos de informática y quiere otorgar de una mayor funcionalidad a la aplicación. Como se ve en el ejemplo de abajo, el usuario puede incluso definir los parámetros que recibirá de los servicios que él elija y su nombre. Tras esto, inserta el código Java que deseé. En este ejemplo si el primer valor es mayor que el segundo, realiza una inserción en la base de datos para dejar un mensaje al móvil. Este ejecutará la acción 0, que es la vibración y le pasa como parámetro el 4000, que se corresponde con 4 segundos en el caso de la vibración.

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

#### 6.4.4 Service Generation

Como se visualiza en la Ilustración 15;**Error! No se encuentra el origen de la referencia.**, esta segunda capa se compone de dos subprocessos. A esta capa le llega en formato XML el DSL creado con el editor web. Aquí, el generador de aplicaciones recibe el DSL en formato XML con la información de la aplicación a generar. El DSL contiene toda la información acerca del flujo que debe seguir el servicio que desea crear el usuario, así como todos los datos que el servicio debe usar. Lo que hace es procesar todos los nodos de información que contiene, creando un árbol con dicha información. Una vez ya tiene el árbol generado, lo recorre y va generando la aplicación con la información que contiene cada nodo. Los nodos

pueden ser condiciones, instrucciones o acciones. Cada uno, con su propia configuración para definir lo que realizará. De esta manera, dos nodos del mismo tipo, además de estar situados en diferentes partes del código, pueden realizar dos cosas totalmente diferentes. Tras recorrer todo el árbol y generar la aplicación, en nuestro caso una aplicación Java, la compila y la ejecuta en el servidor.

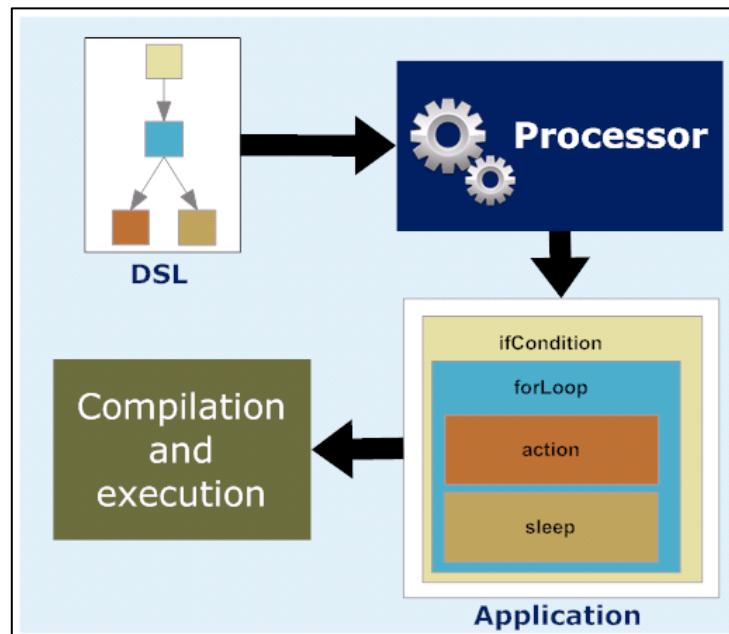


Ilustración 15 Funcionamiento interno del Service Generation

#### 6.4.5 Processor and Object Manager

Esta capa está formada por dos módulos que trabajan de forma conjunta: el **servidor de procesos** y el **almacén de datos**.

**Servidor de procesos:** En esta capa también se encuentran las aplicaciones realizadas por los usuarios. Éstas están siendo ejecutadas continuamente mientras dure su ciclo de vida y están realizando sus peticiones de información contra el servidor. Realizan consultas continuamente a la base de datos y, cuando se cumple lo estipulado por el usuario en ellas, introducen el mensaje que debe enviar el servidor al Smart Object e insertan una notificación para avisar al servidor de la existencia de respuestas para dicho objeto.

**El almacén de datos.** Este se encarga de centralizar todas las peticiones, registrar los objetos inteligentes junto a sus servicios, guardar todos los datos y acciones de los servicios y de servir a las aplicaciones creadas por los usuarios de dicha información. Para ello hace uso de una arquitectura orientada a servicios, al igual que sugieren en otras investigaciones (Atzori et al., 2010; Gama et al., 2012; Pintus et al., 2011, 2010) en conjunto a una arquitectura REST, como sugieren en (Guinard et al., 2009) por sus beneficios y su facilidad de uso. De esta forma, se consigue que sea accesible para

Caso de estudio: MIDGAR

---

cualquier objeto que implemente el mismo lenguaje de mensajes que la plataforma MIDGAR. De esta manera, y mediante el uso de REST los objetos reciben y envían todos los mensajes necesarios. Cuando recibe un mensaje de un objeto, lo procesa y guarda los datos en la base de datos. Después comprueba si hay mensajes pendientes para dicho objeto. Si los hay, genera la respuesta con todos los mensajes pendientes para el Smart Object y se lo envía como respuesta a su petición. En caso contrario, le envía un mensaje de confirmación que indica la recepción correcta de la información.

#### 6.4.6 *Objects*

Los objetos o Smart Objects que quieran conectarse con MIDGAR deben implementar una especificación de mensajes entendible por el servidor. Esta especificación define la interfaz por el cuál un objeto debe presentar sus acciones y servicios. Lo primero que debe hacer un objeto es registrarse en la plataforma. Así las acciones y servicios del objeto se registran en el almacén de datos de MIDGAR para que posteriormente los usuarios puedan utilizar los servicios de los objetos registrados en la especificación de procesos de coordinación entre objetos. Los servicios que puede ofrecer un objeto son aquellos datos que puede proporcionar. Por ejemplo, un Smartphone puede proporcionar un servicio por sensor, un coche podría dar el kilometraje o el gasto de combustible y la centralita de una Smart Home podría dar la temperatura de cada habitación, su humedad o el estado de las ventanas o puertas (abiertas o cerradas). Mientras, las acciones son lo que puede realizar cada objeto. Un móvil podría vibrar durante un determinado periodo de tiempo, mandar un SMS, realizar una llamada o crear una notificación y una Smart Home podría ofrecer el abrir y cerrar puertas y ventanas o encender o apagar la calefacción o el humidificador. A continuación se muestra un ejemplo en el que registra un móvil con el servicio del acelerómetro y la acción de vibrar.

```
<registro>
    <dispositivo>
        <dispositivo_id>e6644a22aae2cec6</dispositivo_id>
        <dispositivo_descripcion>Es el Nexus
4</dispositivo_descripcion>
    </dispositivo>
    <servicio>
        <idservicio>e6644a22aae2cec6-0</idservicio>
        <descripcion>Acelerometro</descripcion>
        <enviaTipo>float</enviaTipo>
    </servicio>
    <action>
        <idaction>e6644a22aae2cec6-0</idaction>
        <action_name>Vibracion</action_name>
        <action_description>Vibracion</action_description>
        <has_message>int</has_message>
    </action>
</registro>
```

Una vez está registrado, el objeto puede comenzar a enviar datos. En cada envío, adjunta todos los datos de los servicios existentes: el dato a enviar y el número del servicio.

```
<envio>
  <datos>
    <dato>8.365426</dato>
    <servicio>e6644a22aae2cec6-0</servicio>
  </datos>
</envio>
```

Tras ello, espera por una respuesta del servidor, ya sea, la confirmación de llegada o un mensaje con acciones que debe ejecutar. En el siguiente ejemplo se puede ver como recibe la petición para realizar la acción 0 con un valor de 1000. En este caso se corresponde con la vibración, a la que le ordena que vibre durante 1 segundo

```
<answer>
  <answer>
    <action>0</action>
    <result>1000</result>
  <answer>
</answer>
```

Para esta propuesta, se utilizó como Smart Object diversos móviles con diferentes versiones del sistema operativo Android y como objeto un micro-controlador Arduino.

#### 6.4.6.1 Prototipo Android

La aplicación nativa hecha para móviles Android da soporte a partir de la versión 2.1, que se corresponde con la API 7. Esta muestra una lista con los sensores poseídos por el teléfono que está siendo utilizado y los valores que capturan en tiempo real. El proyecto permite modificar de una manera sencilla los sensores que se desean enviar, así como escribir el mensaje de registro de los servicios y acciones que posee el dispositivo móvil. Tras esto, se despliega la aplicación en el dispositivo móvil, se selecciona el botón registro y se arranca para que comience el envío de datos al almacenamiento de datos. Estos serán los que utiliza el servidor de procesos para ejecutar las acciones definidas en los flujos de los programas desarrollados por los usuarios con el editor gráfico.

#### 6.4.6.2 Prototipo Arduino

El micro-controlador Arduino (Ilustración 16) utilizado se conectó mediante conexión USB al ordenador. Para realizar el envío y recepción de mensajes se utilizó una aplicación java que hace de intermediaria entre el Arduino y el trato de parámetros. De esta manera, se consigue facilitar el uso de este al no tener que programar ni tener que usar las librerías en C. Además, facilita la creación de aplicaciones sobre Arduino al tener métodos muy parecidos a la aplicación Android y usar un lenguaje de más alto nivel. También permite dotar de una manera fácil de inteligencia al Arduino para así, convertirlo en Smart Object cuando convenga durante las pruebas.

Para su utilización, primero hubo que realizar el montaje de los sensores sobre el micro-controlador y después realizar la configuración de estos en C, utilizando el IDE que posee el propio Arduino y cargarle el programa. Una vez realizado esto, se ejecuta la aplicación java y queda listo para usarse. Primero se registra y después, automáticamente, comienza el envío y recibo de datos.

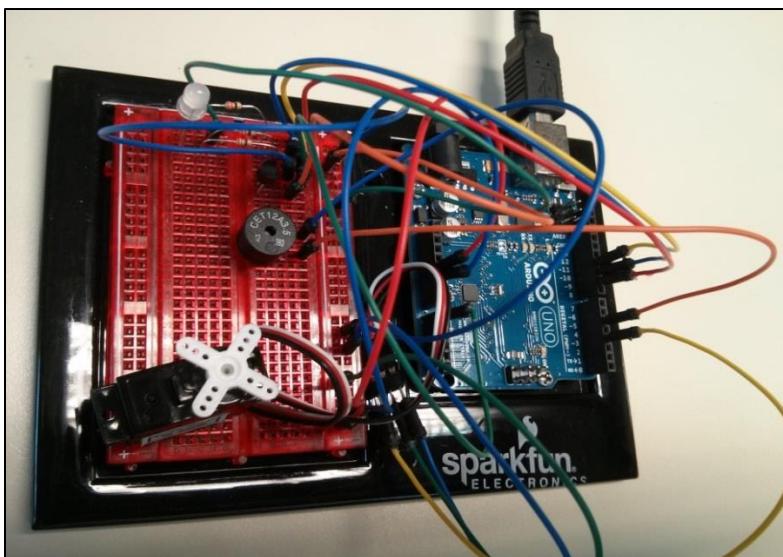


Ilustración 16 Foto del Arduino durante una de las pruebas

## 6.5 *Software utilizado*

Para la realización de dicha investigación se utiliza muy diverso hardware y software:

- El servidor de la investigación se creó utilizando el framework Ruby on Rails.
- El editor se realizó utilizando HTML5 y Javascript.
- Para la creación del generador de aplicaciones y la aplicación de conexión con el Arduino se optó por utilizar Java6. De esta manera, se consiguió dotar de la flexibilidad multiplataforma que pudiese ser necesaria en ambos casos.

Para las pruebas se utilizó el siguiente equipo:

- Servidor alojado en un ordenador dedicado con sistema operativo Windows Server 2008R2 de 64 bits, procesador Intel Core i3-2100 a 3100MHz y 4GB de memoria RAM.
- Servidor web Webrick.
- Base de datos SQLite.
- Cuatro Smartphone: un Nexus 4 con Android 4.2.2, un Motorola con la versión 2.2.2, un Samsung Galaxy S con la versión 2.1 y un Samsung Galaxy Mini S5570 con la versión 2.3.6.
- Micro-controlador Arduino Uno.

## 7 EVALUACIÓN Y DISCUSIÓN

Se hicieron dos pruebas para validar la hipótesis. La primera consta de una toma de datos a los participantes mientras creaban la aplicación interconectora de servicios con diferentes aplicaciones. De esta manera se buscó comprobar la hipótesis de que por medio de un editor gráfico que ofrezca una mayor abstracción respecto al problema, un usuario puede crear de una manera fácil y rápida aplicaciones que interconecten objetos heterogéneos utilizando los mecanismos incluidos en la propuesta.

En la segunda prueba se evaluó la facilidad de creación de aplicaciones que interconecten objetos heterogéneos por medio del editor gráfico. Para ello, se optó por realizar una encuesta a los participantes en las pruebas. Este método es utilizado en el campo de la ingeniería de software para proporcionar información para poder apoyar la efectiva toma de decisiones (Kasunic, 2005). Esto se corresponde a nuestro caso debido a la incapacidad para medir la eficacia de la generación de aplicaciones con el editor gráfico, así como su potencial. Por ello, se espera, que mediante estas encuestas se proporcione la percepción de si el trabajo realizado en esta investigación sirvió para cumplir los objetivos planteados.

En esta sección se presenta detalladamente el experimento realizado y se muestran los resultados obtenidos. En la primera parte de la evaluación se describirá a la metodología utilizada para la realización de las pruebas. Tras esto, se mostrarán y discutirán los resultados obtenidos.

### 7.1 *Metodología*

En esta subsección se explica la metodología seguida para cada prueba: la toma de datos y la encuesta. En ambos apartados se explica con el más mínimo detalle todas las decisiones adoptadas para la realización de ambas pruebas.

#### 7.1.1 *Toma de datos*

Para la realización de las pruebas de medición de tiempos junto a otros datos, se optó por utilizar las siguientes aplicaciones:

- Notepad++ con el plugin XML Tools (Ho, 2013): Este editor de texto enriquecido junto con el plugin, facilitan la creación de XML al ofrecer resaltado de sintaxis, creación de las etiquetas de cierre de un nodo, identado automático y comprobación de la correcta escritura del XML realizado. En este editor hay que escribir todo el código, salvo las etiquetas de cierre.
- Editor Web Textual del proyecto MIDGAR: La primera versión del editor creado para la finalidad de este proyecto fue un editor textual. Tras comprobar y analizar que sería mejor ofrecer un editor gráfico, este se relegó. Este editor cuenta con la inserción automática de código de cada nodo por medio de un botón. De esta manera, lo único que hace falta es saber dónde hay que insertar cada nodo y llenar sus atributos.

- Editor Web Gráfico del proyecto MIDGAR: Como se comentó en el punto 6.4.2, este editor facilita la tarea de creación al usuario. Lo que debe hacer es seleccionar la caja correspondiente al flujo que desea añadir al programa y enlazarla al resto el uso de los conectores.

Para la toma de tiempos se utilizó el programa *Mousotron* (Blacksun Software, 2013). Este permite tomar el tiempo que lleva activo, número de pulsaciones en teclado y en cada botón del ratón, incluyendo las dobles pulsaciones, la distancia recorrida por este y las coordenadas (X e Y) y la velocidad en Km/h. Durante las pruebas se tomó medida de los siguientes factores:

- Tiempo: se midió el tiempo que el participante tardó en realizar la aplicación. Este dato sirve para saber el tiempo que tardó cada participante y averiguar cuál sería la media para compararlo con el tiempo de creación utilizando otras herramientas.
- Pulsaciones de teclas: se contaron las pulsaciones de teclas que hizo el participante. Con este dato se pretende sacar el número de pulsaciones que debe realizar un participante, pues, al equivocarse puede que deba de realizar más, así como al no recibir ayuda de ningún tipo. Esto último da como consecuencia el aumento de la probabilidad de realizar un mayor número de errores.
- Errores: Los errores cometidos por el participante durante la prueba. Entre estos se contabilizaban:
  - Equivocación con los anidamientos de nodos: Cuando un usuario anidaba incorrectamente un nodo dentro de otro.
  - Olvido de parámetros: Cuando se olvidaban de insertar datos e los parámetros de los nodos.
  - Olvido de la etiqueta “*serviceId*:” A veces se olvidaban de insertar esta etiqueta para pedir el parámetro a un objeto en el nodo “*if*” o “*for*”.
  - Inserción de la etiqueta “*serviceId*:” en sitios no requeridos: Otras veces insertaban esta etiqueta en el nodo “*action*”.
  - Desorientación en la creación de la aplicación: A veces los usuarios se perdían y no sabían dónde debían seguir rellenando o insertando datos debido a la cantidad de texto que se encontraba en su pantalla.
  - Erratas: Muchos usuarios cometían erratas a la hora de escribir los nodos o sus atributos, así como utilizaban mayúsculas en vez de minúsculas o se les olvidaba el bloqueo de mayúsculas pulsado.
  - Olvido de datos debido al excesivo texto en pantalla: Algunos usuarios, debido al exceso de texto, no se daban cuenta de que les faltaba por llenar algunos atributos.
- Consultas: A veces, algunos usuarios realizaban alguna consulta durante el transcurso de la prueba debido a que no sabían cómo continuar o se les olvidaba algún paso o acción a realizar o bien cometían un error que no sabían solucionar.

### 7.1.2 Encuesta

Como método de medición para la encuesta, se utilizó la escala Likert (Likert, 1932). Se usó esta por ser la medida más utilizada en el diseño de escalas. Se optó por usar el *5-points Likert Scale* dando como opciones: 1 como *Totalmente en desacuerdo*, 2 como *En desacuerdo*, 3 como *Neutral*, 4 como *De acuerdo* y 5 como *Totalmente de acuerdo*.

Para la muestra se eligió un tamaño que fuera representativo de la población y que pudiesen otorgar resultados significativos al estudio (Kitchenham & Pfleeger, 2002). Como perfiles se decidió elegir desarrolladores de software (SDev) y usuarios interesados en IoT (IoTU). Se eligió estos dos perfiles en base a la segunda contribución: conseguir una capa de abstracción sobre el DSL que proporcione la facilidad de creación de las aplicaciones a cualquier tipo de usuario, sea o no desarrollador de software.

Para realizar el muestreo de la población se utilizó el método bola de nieve (Groves et al., 2004). Para ello, se partió de gente conocida. Estos fueron invitando a otra gente a participar en la prueba. En total fueron 21 participantes. 12 eran SDev y 9 eran IoTU. De esta manera se esperaba poder evaluar el editor y el generador de aplicaciones desde ambos puntos de vista: desarrollador y usuario. Esto se debe a que cada uno aprecia y busca una cosa diferente a pesar de que deban realizar la misma tarea.

Para la realización de las pruebas ambos perfiles debían utilizar primero el Notepad++, después el editor textual y al final el editor gráfico. Durante todo el procedimiento se procedió con los participantes individualmente. Primero se les explicó el escenario del proyecto, sus problemas actuales y la investigación. Después se les explicaba la aplicación que debían realizar sobre un caso real. De esta manera se buscó un mejor entendimiento por parte de los usuarios. En ella debían de comprobar la temperatura del micro-controlador Arduino. Cuando este superase los 25°C debía de enviar una vibración de 2500 milisegundos al Nexus 4 por cada grado centígrado sobrepasado y una notificación de aviso al Motorola.

Una vez lo comprendían, se les mostraba la aplicación que debían utilizar. Para las tres aplicaciones se les explicó: colocación de botones, significado, distribución y uso. Tras este procedimiento realizaban la aplicación. Una vez acababan, se cambiaban de ordenador y rellenaban, de manera anónima, sin ayuda y en privado el cuestionario (Tabla 2).

Pregunta	Descripción
<b>Q1</b>	Esta herramienta ofrece una asistencia útil como editor de desarrollo para la creación de aplicaciones que interconecten objetos.
<b>Q2</b>	Este editor ofrece una forma eficaz de desarrollar la tarea indicada.
<b>Q3</b>	La forma de crear aplicaciones mediante el uso del editor es comprensible.
<b>Q4</b>	El uso del editor disminuye la complejidad de desarrollo de este tipo de aplicaciones.
<b>Q5</b>	El editor ofrece herramientas suficientes para crear cualquier tipo de aplicación que interconecte objetos.
<b>Q6</b>	Basado en su experiencia previa con el uso de otras herramientas, este editor ofrece una mayor facilidad para la generación de aplicaciones de este tipo.
<b>Q7</b>	Internet of Things y los Smart Objects podrían beneficiarse de esta herramienta al contribuir con una gran facilidad para interconectar objetos.
<b>Q8</b>	Este editor proporciona una forma fácil e intuitiva para interconectar dispositivos.
<b>Q9</b>	Este editor podría ser enfocado al desarrollo de otro tipo de aplicaciones similares.
<b>Q10</b>	El usuario comete menos errores mientras trabaja de una manera más rápida y eficaz mediante el uso de este editor.
<b>Q11</b>	Este editor puede contribuir a facilitar la aparición de servicios que ofrezcan interconexión de objetos.
<b>Q12</b>	Este editor puede considerarse útil y usable.

**Tabla 2 Cuestionario realizado a los usuarios**

Para la realización del cuestionario se buscaron un total de doce declaraciones. En ellas se preguntaba acerca de la opinión sobre el uso herramienta, sus posibilidades y su posible impacto en Internet of Things y los Smart Objects.

Para la evaluación de los resultados, se optó por mostrar las respuestas de los usuarios por separado pero evaluarlas en conjunto. Esto se debe a que interesaba evaluar desde el punto de vista de ambos perfiles. No obstante, lo que se busca es ofrecer la misma herramienta y funcionalidad a ambos, por eso se evalúa en conjunto. Por ello, esta es la primera evaluación que hacemos. A continuación, se hacen la de ambos perfiles por separado.

Los datos obtenidos para la evaluación son variables estadísticas cuantitativas, ya que se expresan mediante valores numéricos. Además estas son continuas ya que se sitúan dentro de un rango de valores determinado por la escala Likert de cinco puntos (1 a 5). Para evaluar los datos recogidos se optó por usar las siguientes medidas:

- Medidas de centralización: Con un único valor se identifica la tendencia central de la variable.
  - Moda: Es el valor de la variable que aparece con más frecuencia en la muestra. Puede haber más de una moda. Con este valor se puede obtener cuál es la respuesta más elegida por los usuarios respecto a una declaración.
  - Mediana: Es el valor que deja igual número de individuos con valores inferiores o iguales a él que superiores. Con esta medida podemos averiguar cuál es la respuesta de la población que se sitúa en medio de todas. Esto nos da una visión general de hacia dónde apuntan las respuestas.

- Medidas de posición: Determinan la posición que ocupa un individuo dentro de los valores que toma la variable.
  - Cuartiles: Estos son los tres valores que dividen un conjunto de datos ordenado en cuatro partes iguales. Los cuartiles nos sirven para poder valorar las respuestas de los usuarios divididas en cuatro porciones. Esto nos proporciona una mejor información respecto a la diferencia de opiniones entre usuarios.
  - Cuartil 1: Se corresponde con la mediana de la primera mitad de los valores. Este dato ayudará a comparar la opinión de la primera mitad de valores.
  - Cuartil 2: Se corresponde con la mediana.
  - Cuartil 3: Se corresponde con la mediana de la segunda mitad de los valores. Este dato ayudará a comparar la opinión de la segunda mitad de valores.
- Medidas de dispersión: Miden el grado de variabilidad de los datos muestrales y se usan para completar la información que dan las medidas de centralización, midiéndonos la representatividad de los valores centrales.
  - Rango: Es la diferencia entre los valores de los extremos. Con él podemos saber cuánta diferencia hay entre las respuestas extremas. Esto nos indicará si hay mucha diferencia en la opinión de los usuarios.
  - Rango Inter-Cuartílico (RIC): Es la diferencia entre el primer y tercer cuartil. Con este dato podremos saber la diferencia entre las medianas de ambos grupos de cuartiles.
- Otras:
  - Mínimo: Número menor de los datos. Con este dato podremos saber cuál fue la calificación más baja dada por un participante a una declaración.
  - Máximo: Número mayor de los datos. Con este dato podremos saber cuál fue la calificación más alta dada por un participante a una declaración.

Para la visualización de los datos se optó por utilizar un diagrama de cajas y bigotes que ayuda a visualizar los datos antes descritos y así poder comparar todas las declaraciones a la vez. La Ilustración 17 es un ejemplo donde se ve dicho diagrama y como se corresponde las medidas que se utilizarán.

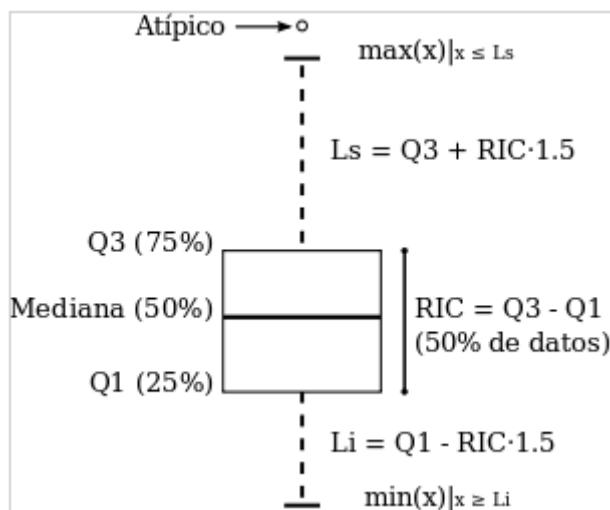


Ilustración 17 Ejemplo de un diagrama de cajas y bigotes

## 7.2 *Resultados de la toma de tiempos*

En la Tabla 3, Tabla 4 y Tabla 5 se muestran los datos recogidos a todos los participantes durante la realización de las pruebas. Estos se corresponden con la creación de la aplicación requerida con el Notepad++ con el Plugin XML Tools, el editor textual y el editor gráfico, respectivamente. En cada tabla se muestra el identificador del participante, el tiempo, en segundos que tardó en hacerlo, las pulsaciones de teclado de hizo, los errores cometidos y el número de consultas que hizo al supervisor durante la prueba.

<b>Id</b>	<b>Tiempo (seg.)</b>	<b>Pulsaciones de teclado</b>	<b>Errores</b>	<b>Consultas</b>
<b>p1</b>	627	722	8	2
<b>p2</b>	403	468	5	0
<b>p3</b>	429	649	5	1
<b>p4</b>	245	494	3	0
<b>p5</b>	270	437	3	0
<b>p6</b>	555	449	2	1
<b>p7</b>	461	573	4	2
<b>p8</b>	281	504	1	0
<b>p9</b>	625	564	5	2
<b>p10</b>	673	489	7	0
<b>p11</b>	600	501	6	0
<b>p12</b>	360	582	6	0
<b>p13</b>	803	584	11	3
<b>p14</b>	344	443	5	1
<b>p15</b>	235	495	4	1
<b>p16</b>	857	796	5	0
<b>p17</b>	344	457	3	0
<b>p18</b>	454	478	4	1
<b>p19</b>	280	554	8	0

Tabla 3 Toma de datos en la realización de la aplicación con el Notepad++

<b>Id</b>	<b>Tiempo (seg.)</b>	<b>Pulsaciones de teclado</b>	<b>Errores</b>	<b>Consultas</b>
<b>p1</b>	267	250	1	1
<b>p2</b>	197	132	2	0
<b>p3</b>	263	191	5	1
<b>p4</b>	152	209	0	1
<b>p5</b>	144	161	2	0
<b>p6</b>	195	188	2	0
<b>p7</b>	186	173	2	1
<b>p8</b>	145	159	1	0
<b>p9</b>	297	317	1	0
<b>p10</b>	242	93	2	0
<b>p11</b>	309	129	3	0
<b>p12</b>	185	163	2	1
<b>p13</b>	280	86	5	1
<b>p14</b>	159	139	1	0
<b>p15</b>	134	208	2	0
<b>p16</b>	231	248	0	0
<b>p17</b>	147	98	2	0
<b>p18</b>	145	123	4	0
<b>p19</b>	113	113	2	0

Tabla 4 Toma de datos en la realización de la aplicación con el Editor Textual

<b>Id</b>	<b>Tiempo (seg.)</b>	<b>Pulsaciones de teclado</b>	<b>Errores</b>	<b>Consultas</b>
<b>p1</b>	305	30	1	0
<b>p2</b>	165	24	0	0
<b>p3</b>	177	24	1	1
<b>p4</b>	117	24	0	0
<b>p5</b>	131	26	2	0
<b>p6</b>	185	28	1	1
<b>p7</b>	155	32	0	1
<b>p8</b>	99	22	1	0
<b>p9</b>	169	24	0	0
<b>p10</b>	225	20	0	0
<b>p11</b>	203	20	1	0
<b>p12</b>	142	23	1	0
<b>p13</b>	264	26	2	1
<b>p14</b>	127	26	1	1
<b>p15</b>	82	22	0	0
<b>p16</b>	390	40	0	0
<b>p17</b>	131	21	0	0
<b>p18</b>	141	27	1	0
<b>p19</b>	93	28	1	0

Tabla 5 Toma de datos en la realización de la aplicación con el Editor Gráfico

### 7.2.1 Comparativa de Tiempo

En la Ilustración 18 se puede ver la gráfica que compara el tiempo medio que los participantes han tardado en crear la aplicación pedida en cada editor.

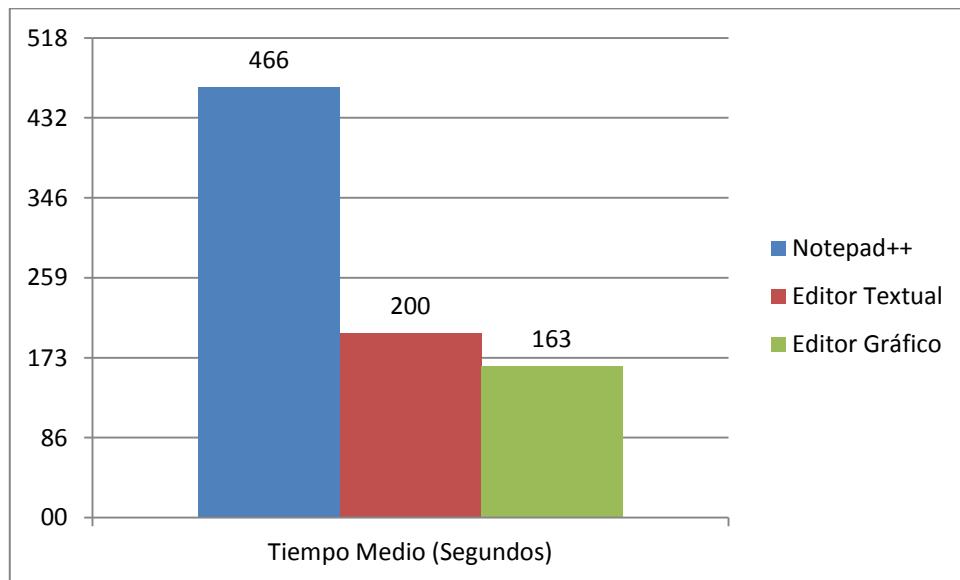


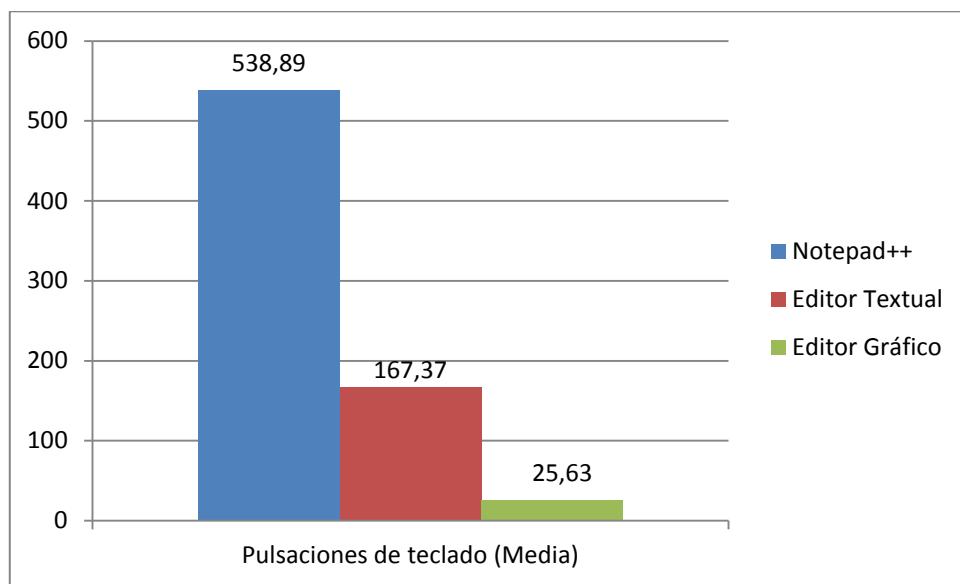
Ilustración 18 Comparativa de tiempos medio de creación de las aplicaciones

Se puede observar como con el Notepad++ tardaron una media de 466 segundos, lo equivalente a poco más de 8 minutos. No obstante, con el editor textual tardaron tres minutos y veinte segundos. Casi cinco minutos menos. Esto indica una gran disminución, pues es más de la mitad del tiempo. Con el editor gráfico tardaron dos casi tres minutos. Una pequeña disminución, exactamente de 37 segundos respecto al editor textual. De esto se puede sacar que los usuarios tardaron menos con este último editor. No obstante, la diferencia entre ambos editores de la plataforma no es tan grande como para decantarse por uno, como si ocurre con el uso del Notepad++ que los supera en más del doble de tiempo.

Estos números se deben, claramente, a la diferencia existente entre tener que escribir casi toda la aplicación por medio del uso del DSL, como sucede con el Notepad++, a sólo tener que escribir los atributos, que es lo que sucede en los dos editores propios. Por ello, tanto el editor textual y el gráfico tienen prácticamente el mismo tiempo, resultando casi irrelevante la diferencia de tiempo medio entre ambos, pues sólo es de 37 segundos. Lo que no es irrelevante es la mejora sustancial en tiempo que ofrecen estos dos editores frente a tener que escribir toda la aplicación.

### 7.2.2 Comparativa de pulsaciones de teclado

En la Ilustración 19 Ilustración 18 se puede ver la gráfica que compara el número medio de pulsaciones de teclas necesitadas por los participantes al crear la aplicación pedida en cada editor.



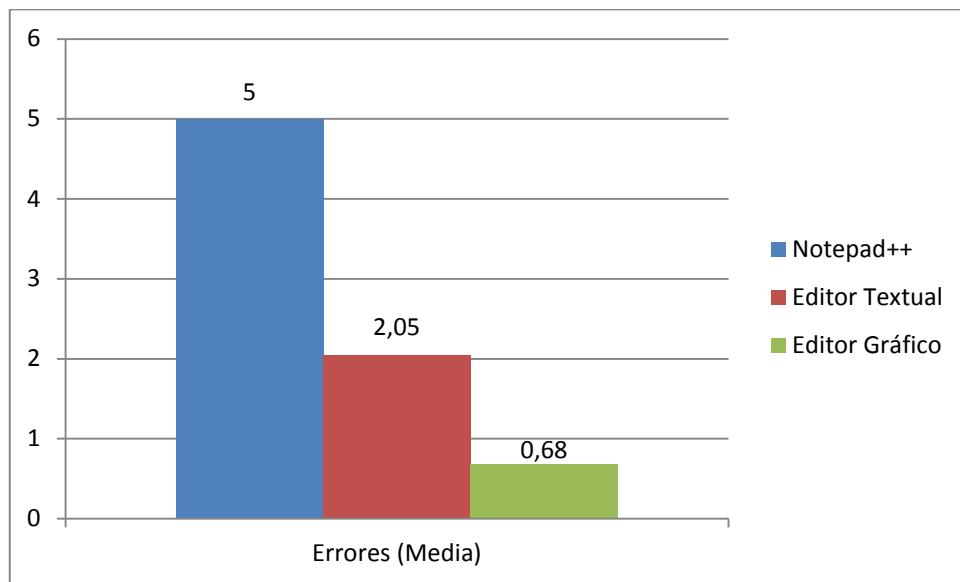
**Ilustración 19 Comparativa con las pulsaciones de teclado medias en la creación de las aplicaciones**

Con el Notepad++ los participantes necesitaron una media de 538,89 pulsaciones. Esto es debido a que debían de escribir todo el código salvo las etiquetas de cierre. Este número de pulsaciones medias se ve drásticamente reducido en 3,2 veces con el uso del editor de texto y, este último, en 7 veces por el editor gráfico. Igual que en el caso del tiempo, el número de teclas pulsadas se ve claramente influenciado por la ayuda que ofrece cada editor: en el caso del Notepad++ solo ayuda con la creación de las etiquetas de cierre; el editor de texto crea el nodo con sus atributos y lo que debe escribir el usuario son los datos que van dentro de cada atributo; en el editor gráfico, el usuario sólo debe escribir algunos atributos, pues otros, simplemente seleccionándolos con el ratón, se los auto-rellena el propio editor.

La interpretación que se puede obtener es la ayuda ofrecida por cada editor a la hora de escribir. Esta diferencia puede repercutir en una disminución en el número de errores al conseguir evitar que el usuario deba teclear partes del lenguaje que pueden insertarse automáticamente.

#### 7.2.3 Comparativa de errores

En la Ilustración 20 Ilustración 19 Ilustración 18 se puede ver la gráfica que compara el número medio de errores cometidos por los participantes al crear la aplicación pedida en cada editor.



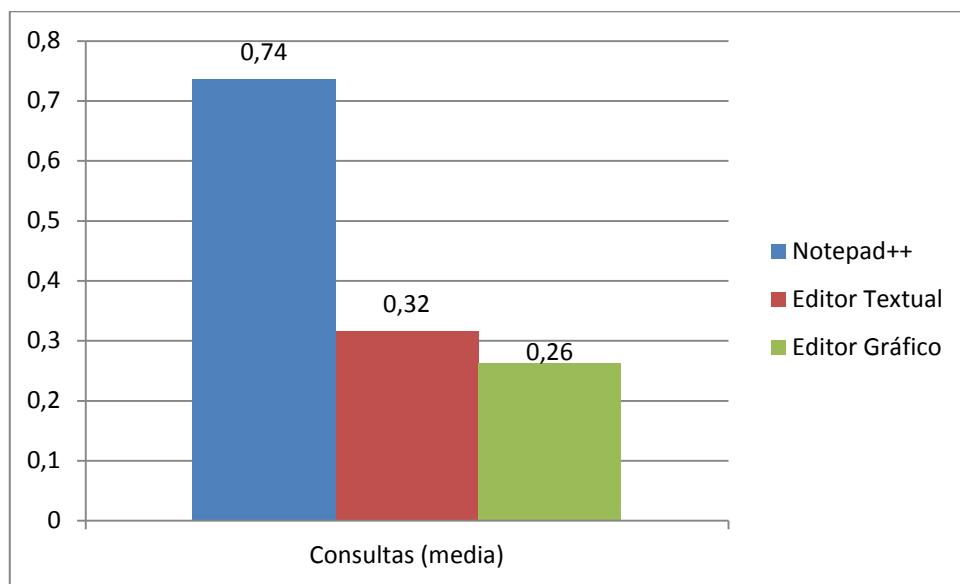
**Ilustración 20 Comparativa con el número de errores medios en la creación de las aplicaciones**

Cuando estos utilizaban el Notepad++ cometieron una media de 5 errores por participante. Con el editor textual, se redujeron la media de los errores en más de la mitad, hasta 2,05. Estos, se reducen algo más de tres veces usando el editor gráfico. Se consigue reducir la media hasta menos de un fallo por persona, exactamente, a un 0,68.

Analizando la gráfica se puede ver como mediante el uso del editor gráfico hay una considerable disminución de los errores respecto al editor de texto, y de este, sobre el Notepad++. Esto va ligado al número de pulsaciones de teclado que deben realizar, pues, los errores analizados, tenían que ver con equivocaciones que cometían al introducir los nodos, sus atributos o los valores. En este caso, mediante el uso del editor gráfico y la ayuda que este les ofrece, se ve cómo repercute en una disminución del número de errores.

#### **7.2.4 Comparativa de consultas**

En la Ilustración 21Ilustración 20Ilustración 19Ilustración 18 se puede ver la gráfica que compara el número medio de consultas realizadas por los participantes al crear la aplicación pedida en cada editor.



**Ilustración 21 Comparativa con el número de consultas medias de los usuarios en la creación de las aplicaciones**

Con el editor gráfico fue con el que menos consultas, un 0,26 de media por persona. No obstante, con el editor textual realizaron pocas más, un 0,31. Estas están muy por debajo de las realizadas con el Notepad++ que suben a algo más del doble, 0,73 consultas por persona.

Al igual que en el caso de los errores, las consultas de los participantes se redujeron en más de la mitad con el uso de los dos editores. Sobre todo en el aspecto de poder seguir correctamente el flujo de creación de la aplicación y no perderse a la hora de crearlo. Como se observa, la diferencia entre el editor de texto y el gráfico es inapreciable, luego, no se puede considerar como un aspecto para destacar uno sobre el otro pero sí, para probar que el uso de un editor que ofrezca ayuda, mejora sustancialmente la creación de la aplicación.

### 7.3 *Resultados de la encuesta*

En la Tabla 6 se puede ver las respuestas enviadas por cada usuario de forma anónima. Esta tabla contiene tanto las respuestas de los participantes que son desarrolladores de software (SDev) como los usuarios interesados en Internet of Things (IoTU).

<b>Id</b>	<b>Perfil</b>	<b>Q1</b>	<b>Q2</b>	<b>Q3</b>	<b>Q4</b>	<b>Q5</b>	<b>Q6</b>	<b>Q7</b>	<b>Q8</b>	<b>Q9</b>	<b>Q10</b>	<b>Q11</b>	<b>Q12</b>
<b>p01</b>	<b>SDev</b>	4	4	3	4	3	4	3	4	4	4	3	3
<b>p02</b>		5	5	5	5	5	5	5	5	4	5	5	5
<b>p03</b>		4	5	4	5	3	5	4	5	3	4	3	4
<b>P04</b>		4	4	3	4	3	4	5	5	4	5	4	4
<b>p05</b>		4	4	3	4	3	4	4	3	4	5	4	3
<b>p06</b>		3	4	4	3	3	5	4	4	3	5	5	4
<b>p07</b>		4	5	3	5	4	4	5	4	5	5	4	5
<b>p08</b>		5	5	4	5	3	5	4	4	5	5	5	5
<b>p09</b>		5	5	4	5	5	5	5	5	4	5	3	4
<b>p10</b>		5	5	4	5	4	5	5	5	5	4	5	5
<b>p11</b>		4	4	5	4	3	5	4	4	5	5	5	5
<b>p12</b>		4	3	3	4	3	3	3	3	3	4	3	3
<b>p13</b>	<b>IoTU</b>	5	4	4	5	4	5	5	4	5	5	4	4
<b>p14</b>		4	5	4	4	4	4	4	4	4	5	4	5
<b>p15</b>		5	5	5	5	5	4	4	4	5	5	5	5
<b>p16</b>		4	4	3	5	3	4	4	3	5	4	5	4
<b>p17</b>		5	4	5	5	5	5	4	5	5	5	4	4
<b>p18</b>		4	5	4	5	4	3	3	4	4	5	4	5
<b>p19</b>		5	5	4	5	5	5	5	5	5	5	4	5
<b>p20</b>		5	5	5	4	5	5	5	5	4	5	5	5
<b>p21</b>		5	5	5	5	5	5	5	5	4	4	5	5

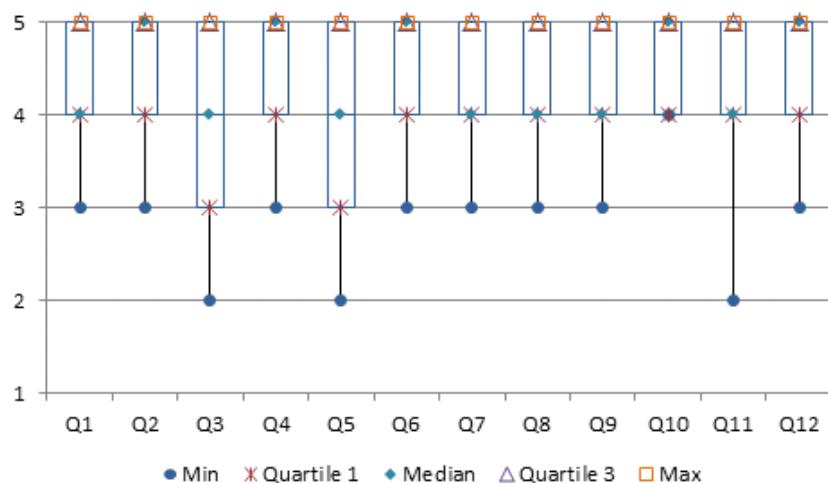
Tabla 6 Respuestas globales de los usuarios en cada pregunta

### 7.3.1 Evaluación global

Primero se evaluará los resultados globales, es decir, las respuestas de los desarrolladores de software y de los usuarios interesados en Internet of Things en conjunto para así obtener la opinión que buscamos. En la Tabla 7 se presentan las estadísticas descriptivas de todo el conjunto. En ella se puede ver, desglosado para cada pregunta, el mínimo, el primer cuartil, la mediana, el tercer cuartil, el máximo, el rango, el rango entre cuartiles y la moda. En la Ilustración 22Figure 5 se pueden ver estos mismos datos en un diagrama de cajas y bigotes.

	<b>Q1</b>	<b>Q2</b>	<b>Q3</b>	<b>Q4</b>	<b>Q5</b>	<b>Q6</b>	<b>Q7</b>	<b>Q8</b>	<b>Q9</b>	<b>Q10</b>	<b>Q11</b>	<b>Q12</b>
<b>Mínimo</b>	3	3	2	3	2	3	3	3	3	4	2	3
<b>Cuartil 1</b>	4	4	3	4	3	4	4	4	4	4	4	4
<b>Mediana</b>	4	5	4	5	4	5	4	4	4	5	4	5
<b>Cuartil 3</b>	5	5	5	5	5	5	5	5	5	5	5	5
<b>Máximo</b>	5	5	5	5	5	5	5	5	5	5	5	5
<b>Rango</b>	2	2	3	2	3	2	2	2	2	1	3	2
<b>Rango Inter-Cuartílico</b>	1	1	2	1	2	1	1	1	1	1	1	1
<b>Moda</b>	4	5	4	5	3	5	5	4	4	5	5	5

Tabla 7 Tabla con las estadísticas descriptivas globales



**Ilustración 22 Diagrama de cajas y bigotes global para cada pregunta**

Analizando la Tabla 7 e Ilustración 22 se pueden obtener las siguientes interpretaciones:

- La Q10 posee el mínimo más alto, en este caso, 4 de un máximo de 5. Esto indica que todos los participantes están, como mínimo, de acuerdo con esta declaración.
- La Q3, Q5 y Q11 poseen el mínimo más bajo, 2 de un máximo de 5. Esto indica que hay participantes *En desacuerdo*.
- La Q2, Q4, Q6, Q10 y Q12 son las declaraciones con la mediana más alta. De esto se puede deducir que la mayoría de los participantes están de acuerdo con estas declaraciones.
- Todas las declaraciones tienen un máximo 5. Esto indica que en todas hubo al menos un participante que estuvo *Muy de acuerdo*.
- La Q10 es la única declaración que posee un rango de 1. Esto indica que todos los participantes tienen prácticamente la misma opinión acerca de esta declaración.
- Las declaraciones Q3, Q5 y Q11 son las que mayor rango poseen. Esto indica que hay una gran diferencia entre las opiniones de los participantes.
- Según la moda, se puede observar que la Q5 posee una moda 3. Esto indica que la respuesta más elegida por los participantes es *Neutral*.
- La Q2, Q4, Q6, Q7, Q10, Q11 y Q12 poseen moda 5, lo que indica que la respuesta más elegida por los participantes en estas declaraciones fue *Totalmente de acuerdo*.
- Respecto a la Q11, a pesar de poseer un rango 3 y un mínimo 2, la mediana se sitúa en 4 y el máximo en 5. Esto indica que, a pesar de poseer algún valor bajo la declaración está bastante bien valorado.

En la Tabla 8 se puede ver la frecuencia de las respuestas para cada pregunta. En esta se puede ver desglosado para cada pregunta el número de votos de cada decisión elegida y su porcentaje

correspondiente para ambos conjuntos. En la Ilustración 23Figure 6 se muestra en un gráfico de barras la frecuencia de respuestas en el conjunto formado por ambos perfiles. En la Ilustración 24Ilustración 30 se muestran las respuestas de estos mediante un gráfico de barras apilado marcando los percentiles

Pregunta		Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Q1	#	0	0	1	10	10
	%	0%	0%	5%	48%	48%
Q2	#	0	0	1	8	12
	%	0%	0%	5%	38%	57%
Q3	#	0	1	5	9	6
	%	0%	5%	24%	43%	29%
Q4	#	0	0	1	7	13
	%	0%	0%	5%	33%	62%
Q5	#	0	2	8	4	7
	%	0%	10%	38%	19%	33%
Q6	#	0	0	2	7	12
	%	0%	0%	10%	33%	57%
Q7	#	0	0	3	9	9
	%	0%	0%	14%	43%	43%
Q8	#	0	0	3	9	9
	%	0%	0%	14%	43%	43%
Q9	#	0	0	3	9	9
	%	0%	0%	14%	43%	43%
Q10	#	0	0	0	6	15
	%	0%	0%	0%	29%	71%
Q11	#	0	1	4	7	9
	%	0%	5%	19%	33%	43%
Q12	#	0	0	3	7	11
	%	0%	0%	14%	33%	52%

Tabla 8 Tabla de frecuencias de las respuestas globales

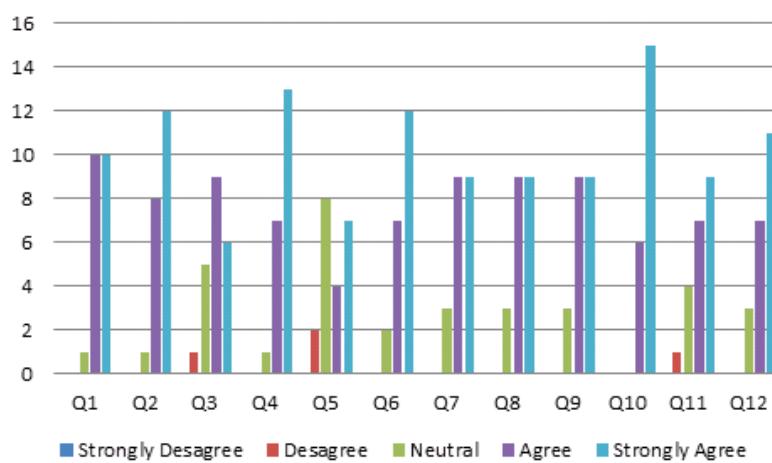


Ilustración 23 Distribución de las respuestas globales

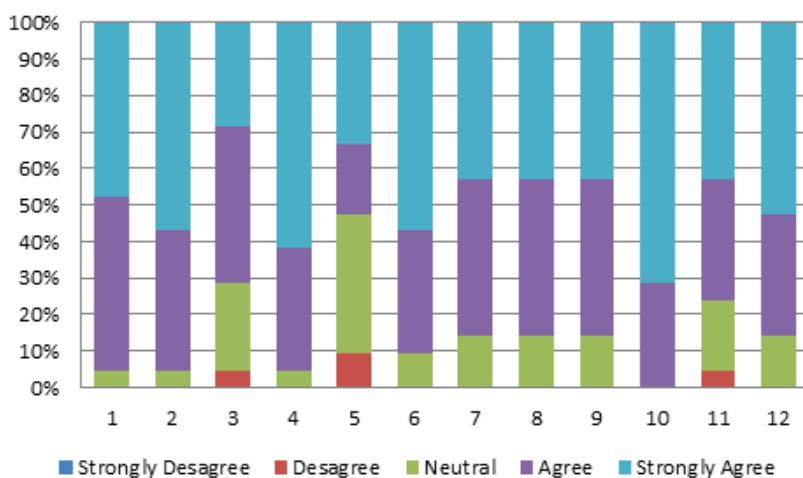


Ilustración 24 Distribución apilada de las respuestas globales

A partir de la Tabla 8, Ilustración 23 e Ilustración 24 se pueden sacar interpretaciones que no se pudieron anteriormente. Estas son las siguientes interpretaciones:

- La Q1, posee un 48% de votación como *De acuerdo* y *Fuertemente de acuerdo*, mientras que solo el 5% votó *Neutral*. En números esto se traduce como 11, 11 y 1 voto respectivamente. Esto indica que los participantes está *De acuerdo* como mínimo, a excepción de una pequeña minoría con esta declaración.
- Las declaraciones Q7, Q8 y Q9, tienen un 43% de votos en *De acuerdo* y *Fuertemente de acuerdo* y un 14% en *Neutral*. La correspondencia en números es, respectivamente, 9, 9 y 3. Esto indica que la mayoría está *De acuerdo* pero hay bastante porcentaje que está indecisa o no cree en esas declaraciones.

### 7.3.2 Evaluación de los desarrolladores de software

En esta sección se evaluarán las respuestas de los desarrolladores de software. Estos participantes están familiarizados con el desarrollo de software y pueden dar unas respuestas desde un punto de vista más técnico que el de un usuario. En la Tabla 9Tabla 7 se presentan las estadísticas descriptivas pertenecientes a los desarrolladores de software. En ella se puede ver, desglosado para cada pregunta, el mínimo, el primer cuartil, la mediana, el tercer cuartil, el máximo, el rango, el rango entre cuartiles y la moda. En la Ilustración 25 se pueden ver estos mismos datos en un diagrama de cajas y bigotes.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
<b>Mínimo</b>	3	3	2	3	2	3	3	3	3	4	3	3
<b>Cuartil 1</b>	4	4	3	4	3	4	4	4	3,75	4	3	3,75
<b>Mediana</b>	4	4,5	4	4,5	3	5	4	4	4	5	4	4
<b>Cuartil 3</b>	5	5	4	5	4	5	5	5	5	5	5	5
<b>Máximo</b>	5	5	5	5	5	5	5	5	5	5	5	5
<b>Rango</b>	2	2	3	2	3	2	2	2	2	1	2	2
<b>Rango Inter-Cuartílico</b>	1	1	1	1	1	1	1	1	1,25	1	2	1,25
<b>Moda</b>	4	5	4	5	3	5	5	4	4	5	5	5

Tabla 9 Estadísticas descriptivas de los desarrolladores de software

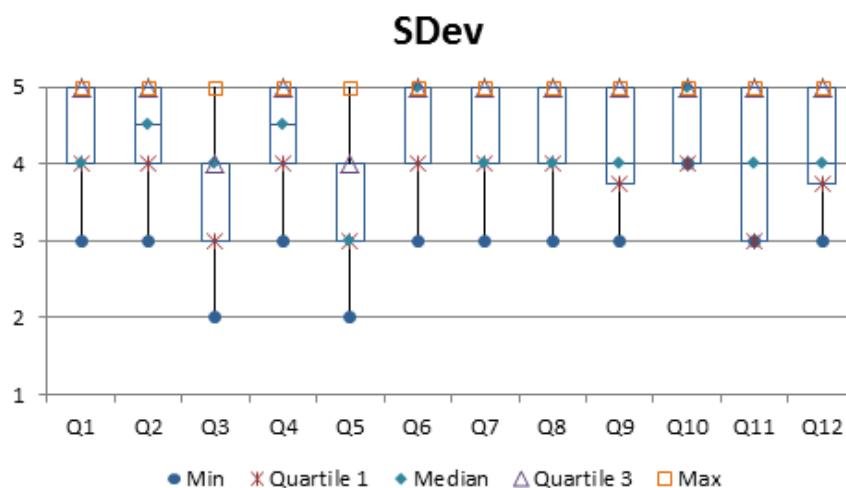


Ilustración 25 Diagrama de cajas y bigotes de los desarrolladores de software para cada pregunta

Analizando la Tabla 9/Tabla 7 e Ilustración 25 se pueden obtener las siguientes interpretaciones:

- La Q10 posee el mínimo más alto, en este caso, 4 de un máximo de 5. Esto indica que todos los participantes están, como mínimo, *De acuerdo* con esta declaración.
- La Q2 y Q5 tienen el mínimo más bajo, 2 de un máximo de 5. Esto indica hay participantes *En desacuerdo*.
- La Q6 y Q10 son las declaraciones con la mediana más alta. De esto se puede deducir que la mayoría de los participantes están *De acuerdo* con estas declaraciones.
- La Q5 tiene la mediana más baja, 3. Esto indica que los participantes no están ni de acuerdo ni en desacuerdo (*Neutral*) con esta declaración.
- Todas las declaraciones tienen un máximo 5. Esto indica que en todas hubo al menos un participante que estuvo *Muy de acuerdo*.
- La Q10 es la única declaración que posee un rango de 1. Esto indica que todos los participantes tienen prácticamente la misma opinión acerca de esta declaración.

- Las declaraciones Q3 y Q5 son las que mayor rango poseen. Esto indica que hay una gran diferencia entre las opiniones de los participantes.
- Según la moda, se puede observar que la Q5 posee una moda 3. Esto indica que la respuesta más elegida por los participantes es *Neutral*.
- La Q2, Q4, Q6, Q7, Q10, Q11 y Q12 poseen moda 5. Esto que indica que la respuesta más elegida por los participantes en estas declaraciones fue *Totalmente de acuerdo*.

En la Tabla 10Tabla 8 se puede ver la frecuencia de las respuestas para cada pregunta. En esta se puede ver desglosado para cada pregunta el número de votos de cada decisión elegida y su porcentaje correspondiente para los desarrolladores de software. En la Ilustración 26Figure 6 se muestra en un gráfico de barras la frecuencia de respuestas en el conjunto formado por el perfil SDev. En la Ilustración 27Ilustración 30 se muestran las respuestas de estos mediante un gráfico de barras apilado marcando los percentiles

Pregunta		Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
<b>Q1</b>	#	0	0	1	7	4
	%	0%	0%	8%	58%	33%
<b>Q2</b>	#	0	0	1	5	6
	%	0%	0%	8%	42%	50%
<b>Q3</b>	#	0	1	4	5	2
	%	0%	8%	33%	42%	17%
<b>Q4</b>	#	0	0	1	5	6
	%	0%	0%	8%	42%	50%
<b>Q5</b>	#	0	1	7	2	2
	%	0%	8%	58%	17%	17%
<b>Q6</b>	#	0	0	1	4	7
	%	0%	0%	8%	33%	58%
<b>Q7</b>	#	0	0	2	5	5
	%	0%	0%	17%	42%	42%
<b>Q8</b>	#	0	0	2	5	5
	%	0%	0%	17%	42%	42%
<b>Q9</b>	#	0	0	3	5	4
	%	0%	0%	25%	42%	33%
<b>Q10</b>	#	0	0	0	4	8
	%	0%	0%	0%	33%	67%
<b>Q11</b>	#	0	0	4	3	5
	%	0%	0%	33%	25%	42%
<b>Q12</b>	#	0	0	3	4	5
	%	0%	0%	25%	33%	42%

**Tabla 10 Tabla de frecuencias de las respuesta de los desarrolladores de software**

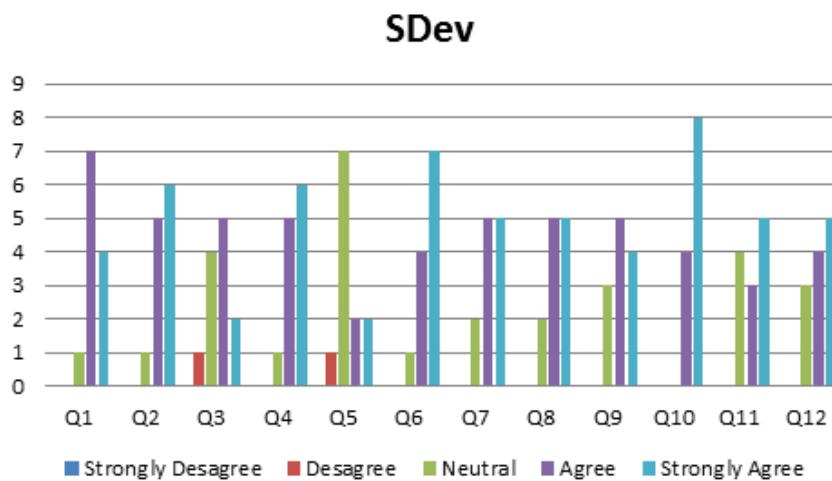


Ilustración 26 Distribución de las respuestas de los desarrolladores de software

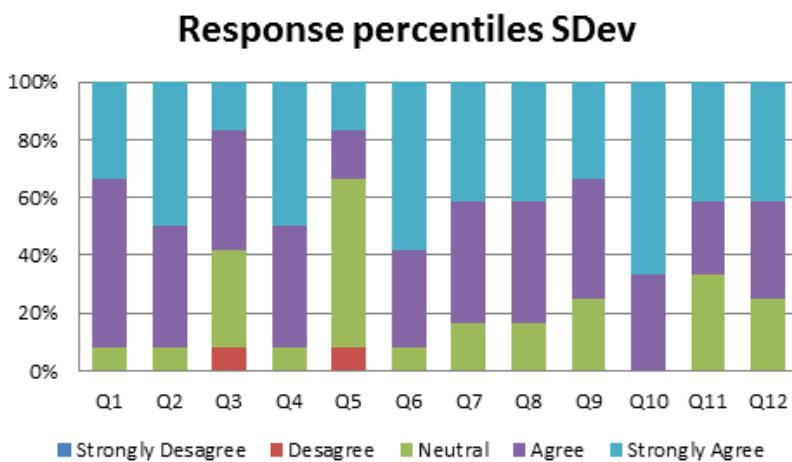


Ilustración 27 Distribución apilada de las respuestas de los desarrolladores de software

A partir de la Tabla 10Tabla 8, Ilustración 26Ilustración 23 e Ilustración 27Ilustración 24 se pueden sacar interpretaciones que no se pudieron anteriormente. Estas son las siguientes interpretaciones:

- La Q1, posee un 58% de votación como *De acuerdo* y un 33% *Fuertemente de acuerdo*, mientras que solo el 8% votó *Neutral*. En números esto se traduce como 7, 4 y 1 voto respectivamente. Esto indica que los participantes está *De acuerdo* como mínimo, a excepción de una pequeña minoría con esta declaración.
- Las declaraciones Q2 y Q4 poseen un 42% de votación como *De acuerdo* y un 50% *Fuertemente de acuerdo*, mientras que solo el 8% votó *Neutral*. En números esto se traduce como 5, 6 y 1 voto respectivamente. Esto indica que los participantes está *De acuerdo* como mínimo, y en su mayoría, *Fuertemente de Acuerdo* a excepción de una pequeña minoría con esta declaración.

- Las declaraciones Q7, Q8 y Q9, tienen un 43% de votos en *De acuerdo* y *Fuertemente de acuerdo* y un 14% en *Neutral*. La correspondencia en números es, respectivamente, 9, 9 y 3. Esto indica que la mayoría está de acuerdo pero hay bastante porcentaje que está indecisa o no cree en esas declaraciones.
- La Q3 parece ser la segunda declaración peor valorada. Esta tiene en total cuatro respuestas diferentes. De más en desacuerdo a más acuerdo, los porcentajes son 8%, 33%, 42% y 17%, que se corresponden en votos con 1, 4, 5 y 2 votos respectivamente. Esto indica que hay bastante desacuerdo respecto a esta declaración, pero, ligeramente los participantes están *De acuerdo*. No obstante, hay 4 *Neutrales* y un *En desacuerdo*, en total, 5 votos no positivos.

### 7.3.3 Evaluación de los usuarios interesados en Internet of Things

En esta sección se evaluarán las respuestas de los usuarios interesados en Internet of Things. Estos participantes conocen la idea de Internet of Things y tienen interés acerca de ello. Esto implica, que esta investigación es de su agrado debido a que se busca facilitar la labor que a ellos les interesa. En la Tabla 11Tabla 7 se presentan las estadísticas descriptivas pertenecientes a los desarrolladores de software. En ella se puede ver, desglosado para cada pregunta, el mínimo, el primer cuartil, la mediana, el tercer cuartil, el máximo, el rango, el rango entre cuartiles y la moda. En la Ilustración 28Ilustración 25 se pueden ver estos mismos datos en un diagrama de cajas y bigotes.

	<b>Q1</b>	<b>Q2</b>	<b>Q3</b>	<b>Q4</b>	<b>Q5</b>	<b>Q6</b>	<b>Q7</b>	<b>Q8</b>	<b>Q9</b>	<b>Q10</b>	<b>Q11</b>	<b>Q12</b>
<b>Mínimo</b>	4	4	3	4	2	3	3	3	4	4	2	4
<b>Cuartil 1</b>	4	4	4	5	4	4	4	4	4	5	4	4
<b>Mediana</b>	5	5	4	5	5	5	4	4	5	5	4	5
<b>Cuartil 3</b>	5	5	5	5	5	5	5	5	5	5	5	5
<b>Máximo</b>	5	5	5	5	5	5	5	5	5	5	5	5
<b>Rango</b>	1	1	2	1	3	2	2	2	1	1	3	1
<b>Rango Inter-Cuartílico</b>	1	1	1	0	1	1	1	1	1	0	1	1
<b>Moda</b>	5	5	4	5	5	5	5	4	5	5	4	5

Tabla 11 Estadísticas descriptivas de los IoTU

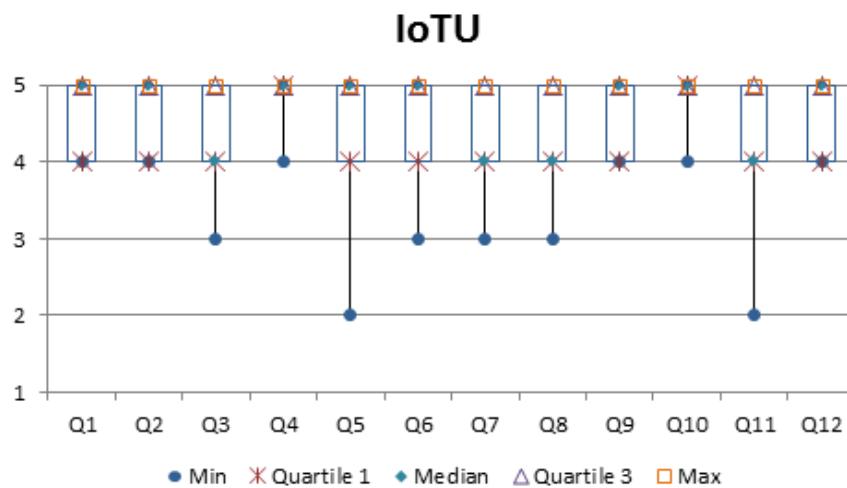


Ilustración 28 Diagrama de cajas y bigotes de los IoTU

Analizando la Tabla 11Tabla 9Tabla 7 e Ilustración 28Ilustración 25 se pueden obtener las siguientes interpretaciones:

- La Q1, Q2, Q4, Q9, Q10 y Q12 poseen el mínimo más alto, en este caso, 4 de un máximo de 5. Esto indica que todos los participantes están, como mínimo, *De acuerdo* con estas declaraciones.
- La Q5 y Q11 tienen el mínimo más bajo, 2 de un máximo de 5. Esto indica hay participantes *En desacuerdo*.
- La Q1, Q2, Q4, Q5, Q6, Q9, Q10 y Q12 son las declaraciones con la mediana más alta. De esto se puede deducir que la mayoría de los participantes están *De acuerdo* con estas declaraciones.
- La Q3, Q7, Q8 y Q11 tiene la mediana más baja, 4. Esto indica que los participantes están como mínimo *De acuerdo* con esta declaración.
- Todas las declaraciones tienen un máximo 5. Esto indica que en todas hubo al menos un participante que estuvo *Muy de acuerdo*.
- La Q1, Q2, Q4, Q9, Q10 y Q12 son las declaraciones que poseen un rango de 1. Esto indica que todos los participantes tienen prácticamente la misma opinión acerca de estas declaraciones.
- Las declaraciones Q5 y Q11 son las que mayor rango poseen. Esto indica que hay una gran diferencia entre las opiniones de los participantes.
- El rango inter-cuartílico de la Q4 y Q10 es 0. Esto significa que la mediana del cuartil 1 y 3 coincide. Indica que casi todos o todos los participantes tienen la misma opinión.
- Según la moda, se puede observar que la Q3, Q8 y Q11 poseen una moda 4. Esto indica que la respuesta más elegida por los participantes es *De acuerdo*.
- La Q1, Q2, Q4, Q5, Q6, Q7, Q9, Q10 y Q12 poseen moda 5. Esto que indica que la respuesta más elegida por los participantes en estas declaraciones fue *Totalmente de acuerdo*.

En la Tabla 12Tabla 10Tabla 8 se puede ver la frecuencia de las respuestas para cada pregunta. En esta se puede ver desglosado para cada pregunta el número de votos de cada decisión elegida y su porcentaje correspondiente para los usuarios interesados en Internet of Things. En la Ilustración 29Ilustración 28Ilustración 26Figure 6 se muestra en un gráfico de barras la frecuencia de respuestas en el conjunto formado por el perfil IoTU. En la Ilustración 30 se muestran las respuestas de estos mediante un gráfico de barras apilado marcando los percentiles.

Pregunta		Totalmente en desacuerdo	En desacuerdo	Neutral	De acuerdo	Totalmente de acuerdo
Q1	#	0	0	0	3	6
	%	0%	0%	0%	33%	67%
Q2	#	0	0	0	3	6
	%	0%	0%	0%	33%	67%
Q3	#	0	0	1	4	4
	%	0%	0%	11%	44%	44%
Q4	#	0	0	0	2	7
	%	0%	0%	0%	22%	78%
Q5	#	0	1	1	2	5
	%	0%	11%	11%	22%	56%
Q6	#	0	0	1	3	5
	%	0%	0%	11%	33%	56%
Q7	#	0	0	1	4	4
	%	0%	0%	11%	44%	44%
Q8	#	0	0	1	4	4
	%	0%	0%	11%	44%	44%
Q9	#	0	0	0	4	5
	%	0%	0%	0%	44%	56%
Q10	#	0	0	0	2	7
	%	0%	0%	0%	22%	78%
Q11	#	0	1	0	4	4
	%	0%	11%	0%	44%	44%
Q12	#	0	0	0	3	6
	%	0%	0%	0%	33%	67%

Tabla 12 Tabla de frecuencias de las respuesta de los IoTU

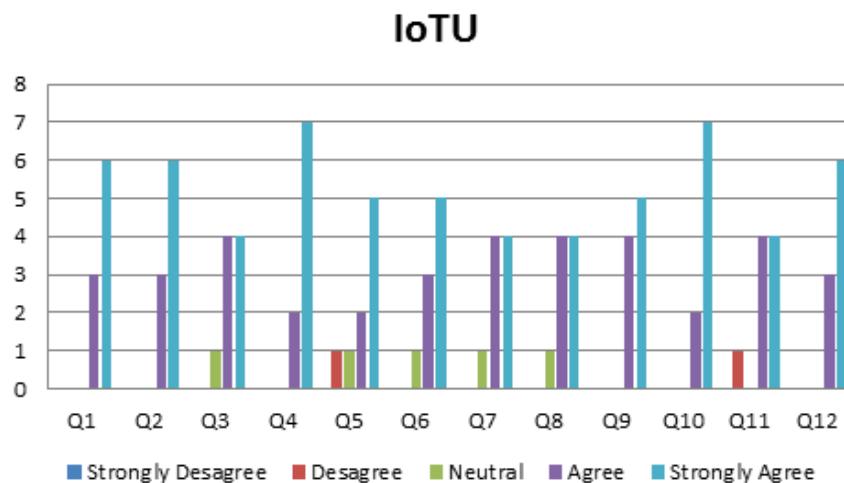


Ilustración 29 Distribución de las respuestas de los IoTU

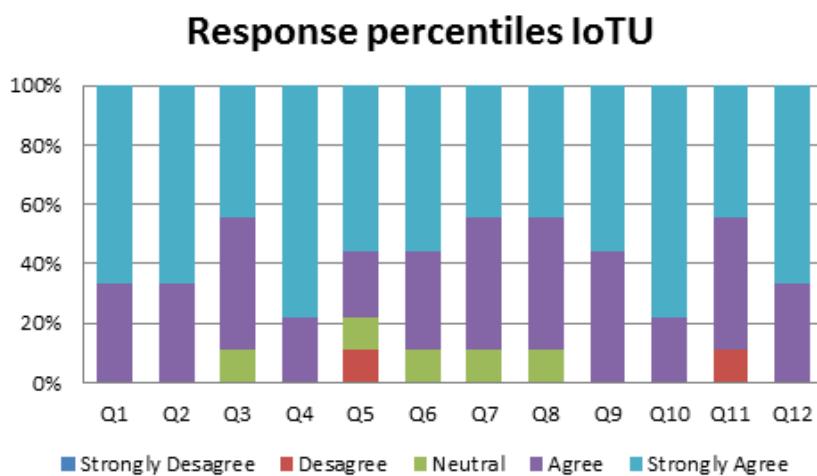


Ilustración 30 Distribución apilada de las respuestas de los IoTU

A partir de la Tabla 12Tabla 8, Ilustración 29Ilustración 28Ilustración 26Ilustración 23 e Ilustración 30Ilustración 27Ilustración 24 se pueden sacar interpretaciones que no se pudieron anteriormente. Estas son las siguientes interpretaciones:

- Las declaraciones Q3, Q6, Q7 y Q8 tienen un 44% de votos en *De acuerdo* y *Fuertemente de acuerdo* y un 11% en *Neutral*. La correspondencia en números es, respectivamente, 4, 4 y 1. Esto indica que la mayoría está *De acuerdo* y sólo una minoría está indecisa o no cree en esas declaraciones.
- La Q5 parece ser la declaración peor valorada. Esta tiene en total cuatro respuestas diferentes. De más en desacuerdo a más acuerdo, los porcentajes son 11%, 11%, 22% y 56%, que se corresponden en votos con 1, 1, 2 y 5 votos respectivamente. Esto indica que hay bastante

acuerdo respecto a esta declaración. No obstante, hay un *Neutral* y un *En desacuerdo*, en total, 2 votos no positivos

- La Q11 es la segunda declaración peor valorada. En ella hay 11% *En desacuerdo*, un 44% *De acuerdo* y un 44% *Fuertemente de acuerdo*. Esto en número de votos se corresponde con 1, 4 y 4 respectivamente. A pesar de ese voto negativo, se puede decir que está bastante bien valorada, pues sólo es un voto no positivo frente a un 88% que están, como mínimo, *De acuerdo*.

## 8 CONCLUSIONES

El objetivo de esta investigación era, mediante la aplicación de Ingeniería Dirigida por Modelos, facilitar la creación de aplicación que permitiesen interconectar objetos heterogéneos a usuarios no expertos. Para conseguir este objetivo, se diseñó un Lenguaje de Dominio Específico que abstrajese este problema y un editor gráfico que facilitase esta tarea. Junto a esto, se creó la plataforma IoT MIDGAR, de forma que consiguiese interconectar estos objetos entre ellos. Por medio de las pruebas realizadas, se consiguió demostrar que el objetivo fue cumplido.

Con la prueba de recogida de tiempo, se ve como la ayuda ofrecida por los editores frente a una pequeña ayuda casi níma del Notepad++, ofrece grandes resultados en la creación de la aplicación utilizando el DSL creado por parte de los participantes. No obstante, entre el editor web textual y el editor web gráfico, los resultados son muy parecidos, a excepción del número de errores cometidos y las pulsaciones de teclado. Estas últimas, claramente, descienden a base de aumentar el uso del ratón. En cuanto a los errores cometidos se observa que con el uso del editor web gráfico estos descienden claramente al permitir que el usuario deba escribir mucho menos gracias al aumentar la abstracción y hacer que la mayoría del texto te lo escriba el propio editor. Así, al conseguir que el usuario cometa menos errores y además, este reduzca un poco el tiempo de creación de la aplicación, se consigue facilitarle el trabajo a realizar.

Según la encuesta realizada, se demuestra que los participantes estaban de acuerdo respecto a que el editor es útil, eficaz y que el usuario comete menos errores y trabaja más rápidamente. También destacan que disminuye la complejidad de desarrollo de este tipo de aplicaciones. No obstante, a pesar de que la mayoría de los participantes están bastante de acuerdo, aún hay algunos que dudan en si el editor podría contribuir a interconectar objetos de una manera fácil e intuitiva y de si se podría aplicar para generar otras aplicaciones similares. Respecto a la creación comprensible de aplicaciones con este editor, así como las herramientas que este proporciona, hay una gran heterogeneidad de respuestas. Sobre si este editor puede facilitar la aparición de servicios que ofrezcan interconexión de objetos, hay también una gran diversidad de opiniones. No obstante, la gran mayoría opina positivamente sobre esta declaración.

El DSL definido es independiente tanto de la plataforma generadora de las aplicaciones como del editor. Por ello, puede ser utilizado como base genérica para crear otros editores y generadores de aplicaciones que interconecten dispositivos heterogéneos o realicen aplicaciones similares. Así mismo, los participantes consideran que con ayuda de un editor gráfico web como el de esta investigación se podría conseguir fomentar la aparición de servicios que ofrezcan la interconexión de objetos. Con esta investigación se pretendió aportar que es posible generar aplicaciones que interconecten objetos heterogéneos de una forma fácil y rápida y que además, sus usuarios no deben por qué ser desarrolladores.

## 9 TRABAJO FUTURO

Internet of Things es parte del futuro y queda mucho por investigar en él. Como se destacó en los problemas, aún hay muchos que resolver. Entre ellos se destacan los que podrían tener solución a partir de esta investigación:

- **Lenguaje de Dominio Específico y editor gráfico para la generación de Objetos Inteligentes:** Creación de un DSL y un editor gráfico, similar a esta investigación, que generé la aplicación que se despliegue en el objeto deseado. Esto facilitaría la creación de Objetos Inteligentes para que se pudiesen interconectar entre ellos. Con él, el usuario sólo deberá de elegir qué servicios quiere ofrecer entre los disponibles en su dispositivo móvil (temperatura, presión, acelerómetro,...) y que acciones ofrecerá (vibración, mensajes, notificaciones, llamadas,...). Durante este proceso, se tendrá la posibilidad de poder definir cierta inteligencia para tratar los datos antes de ser enviados por los servicios y poder modificar ciertas partes de las acciones.
- **Editor de texto con soporte de lenguaje natural:** Mediante esta implementación se espera que cualquier usuario, simplemente describiendo lo que desea, consiga crear la aplicación deseada. Con esto se conseguiría que los usuarios, escribiendo lo que quieren, como si de una búsqueda o un libro se tratase, consiguiesen obtener el resultado deseado. Con ello se obtendría una capa de abstracción que podría facilitar la creación de estas aplicaciones no sólo a usuarios con un mínimo conocimiento informático, si no, a cualquier persona que sepa escribir y redactar. Lo único necesario sería que esta persona posea un conocimiento mínimo acerca de su entorno: los objetos, lo que estos pueden recoger y lo que pueden realizar
- **Interacción y manejo de robots:** Con la aparición de robots ayudantes de personas con discapacidades y problemas, surge la necesidad de que estos puedan ser manejados por estas. Así, en conjunto con el soporte de lenguaje natural, esto último sería posible. La dificultad reside en investigar diferentes tipos de entornos con robots y dotarlos y adaptarlos de todo lo necesario para interactuar como si de otro objeto normal se tratase.
- **Expansión del Lenguaje de Dominio Específico actual para incorporar lógica borrosa:** Por medio de la inclusión de la lógica borrosa se conseguiría que los usuarios pudiesen crear sus propias expresiones. De esta manera, otros usuarios podrían usar estas ya existentes para incorporarlas en el desarrollo de su aplicación.
- **Seguridad y privacidad en Internet of Things:** Estudio sobre las medidas de seguridad y privacidad que debe tener una red Internet of Things para que no se puedan vulnerar los datos trasmitidos desde y para los objetos. IoT busca conectar todos los objetos del mundo y, si hubiese problemas con la seguridad de los objetos o con la privacidad de los usuarios, lo que se conseguiría sería que la gente no lo utilizase al poner en peligro su intimidad. Además, de poder

poner en peligro muchos sistemas que lo utilizasen para automatizar cosas, como pudiera ser en sanidad, industrias químicas, avisos de riesgos ambientales,...

- **Escalabilidad de las plataformas en Internet of Things:** Estudios y pruebas sobre diferentes implementaciones de la plataforma Internet of Things para comprobar con cuál se puede ofrecer una mayor escalabilidad. IoT busca conectar todos los objetos del mundo. Cada persona puede disponer, aproximadamente, de decenas de objetos (relojes, móviles, televisiones, ropa, armarios, neveras,...) y sólo en Europa hay casi mil millones de personas. Esto exige que la red debe soportar una cantidad ingente de tráfico, equiparable o incluso superable al que reciben los servidores de Google o Twitter. Por ello, hay que investigar cuál sería la mejor manera de realizar un sistema escalable y resistente para soportar la red IoT. Un ejemplo de ello es Twitter, que tuvo que migrar de Ruby on Rails a Scala debido a estos problemas.
- **Almacenamiento de datos (SQL vs NoSQL):** Estudio y pruebas sobre qué tipo de base de datos es la mejor para usar en Internet of Things desde el punto de vista del rendimiento y la escalabilidad. Al igual que ocurre con la propia plataforma como se comentó antes, la base de datos es muy importante, pues si esta es lenta y funciona mal, da igual que la plataforma sea escalable. En la actualidad las bases NoSQL son más rápidas que las SQL. No obstante, ciertas puede que tampoco sean la solución debido a la falta de referencias entre los datos, que en este caso, es alta.
- **Soporte de mayor número de dispositivos y sensores:** Ofrecer soporte a un mayor número de dispositivos que dispongan de sensores o la capacidad de conexión de estos.
  - Mayor número de Smartphones: iOS, Windows Phone, Ubuntu for Phones, Firefox OS.
  - Otros micro-controladores: Raspberri Pi.
  - Otros sistemas: GNU/Linux, Os X, PS3, PS4, Xbox, coches,...
  - Integración de Smart Labels: RFID y NFC.
  - Sensores: humedad, proximidad, presencia, potenciómetros, flex sensor, soft potentiometer...
- **Frameworks:** Explorar y desarrollar la mejor solución para permitir que usuarios sin conocimientos técnicos puedan definir servicios y acciones en sus objetos inteligentes para poder conectarlos de una forma fácil y rápida a MIDGAR.

## 10 DIFUSIÓN DE LOS RESULTADOS

### 10.1 *MIDGAR: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios*

Este es el artículo principal de este Trabajo Final de Máster. Trata todo el tema presentado como anteproyecto y contiene gran parte del trabajo realizado, así como una parte de todo lo recogido en este tomo. En este artículo se presenta brevemente la plataforma MIDGAR y se ofrece como contribución el Lenguaje de Dominio Específico creado para desarrollar de una manera fácil y eficaz aplicaciones que interconecten dispositivos en Internet of Things. Se ofrece como apoyo a este DSL un editor gráfico que facilita el desarrollo de las aplicaciones y se evalúa la posibilidad de, por medio de un editor gráfico, facilitar el desarrollo de este tipo de aplicaciones.

Para elegir la revista a la que se enviaría el artículo se hizo un análisis de las revistas existentes. En el análisis se buscó aquellas revistas en las que se consultaron artículos, revistas con artículos del grupo de investigación y otras similares en diferentes editoriales. De este primer análisis, salieron seis posibles revistas pertenecientes al *Journal Citation Report* (Thomson Reuters, 2013):



Ilustración 31 Computer Communications

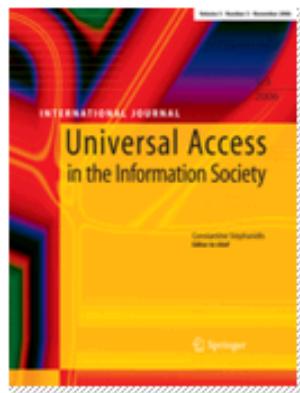
**Computer Communications** (Ilustración 31):

- <http://www.journals.elsevier.com/computer-communications/>
- Editorial: Elsevier
- Índice de Impacto: 1.079
- Índice de Impacto (5 años): 1.227
- Categoría: COMPUTER SCIENCE, INFORMATION SYSTEMS
- Ámbito: Future Internet architecture, protocols and services, Mobile and ubiquitous networks, Internet of things.
- Longitud de los artículos: 14 páginas máximo

Difusión de los resultados

---

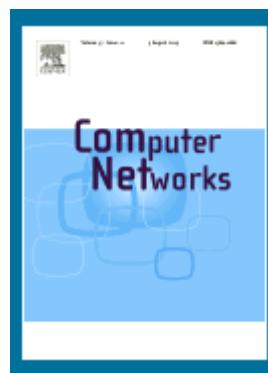
- Esquemas bastante libres y de ámbitos variados (matemáticos, surveys, frameworks, ...)
- Aproximadamente 8 meses desde el envío hasta la publicación digital del artículo



**Ilustración 32 Universal Access in the Information Society**

**Universal Access in the Information Society** (Ilustración 32):

- <http://www.springer.com/computer/hci/journal/10209>
- Editorial: Springer
- Índice de impacto: 0.532
- Índice de Impacto (5 años): -
- Categoría: COMPUTER SCIENCE, CYBERNETICS
- Ámbito: Accesibilidad, usabilidad, facilidad de uso y aceptación de la tecnología de la información ubicua, redes de comunicación, Internet of Things, Smart Objects..
- Longitud de los artículos: 11-15 páginas



**Ilustración 33 Computer Networks**

**Computer Networks** (Ilustración 33):

- <http://www.journals.elsevier.com/computer-networks/>
- Editorial: Elsevier
- Índice de Impacto: 1.231
- Índice de Impacto (5 años): 1.520
- Categoría: COMPUTER SCIENCE, HARDWARE & ARCHITECTURE, INFORMATION SYSTEMS
- Ámbito: Arquitectura, protocolos, servicios y aplicaciones en la red, Internet of Things, Smart Objects.
- Longitud de los artículos: 15-20 páginas

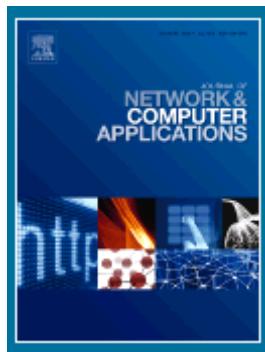


Ilustración 34 Journal of Network and Computer Applications

**Journal of Network and Computer Applications** (Ilustración 34):

- <http://www.journals.elsevier.com/journal-of-network-and-computer-applications>
- Editorial: Elsevier
- Índice de Impacto: 1.467
- Índice de Impacto (5 años): 1.251
- Categoría: COMPUTER SCIENCE, HARDWARE & ARCHITECTURE, INTERDISCIPLINARY APPLICATIONS, SOFTWARE ENGINEERING
- Ámbito: Internet of Things, computer networks, standards for Internet, applications of networked.
- Longitud de los artículos: Aproximadamente 13 páginas
- Aproximadamente 7 meses desde el envío hasta la publicación digital del artículo

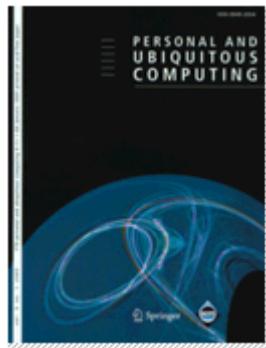


Ilustración 35 Personal and Ubiquitous Computing

**Personal and Ubiquitous Computing** (Ilustración 35):

- <http://www.springer.com/computer/hci/journal/779>
- Editorial: Springer
- Índice de Impacto: 1.133
- Índice de Impacto (5 años): 1.169
- Categoría: COMPUTER SCIENCE, INFORMATION SYSTEMS
- Ámbito: Mobile information appliances, new personal technologies, Industrial and academic research contributions.
- Longitud de los artículos: 10-14 páginas
- Aproximadamente 4-5 meses desde el envío hasta la publicación digital del artículo

Se eligió la revista **Computer Networks** por ser la revista en donde mejor encajaba el artículo en lo referente al ámbito, así como los artículos publicados y la longitud de estos. Además, era la segunda con mayor índice de impacto de las revistas elegidas. Así pues, este artículo se envió el día 03-07-2013 a la revista Computer Networks con un índice de impacto de 1.231 y que pertenece a la editorial Elsevier. Su estado “*Under review*” desde el día 11-07-2013. Ref. No.: COMNET-D-13-1528

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

Esta difusión es un capítulo de libro que presenta la propuesta de este Trabajo Final de Máster (TFM) antes de comenzarlo. El libro trataba sobre Ingeniería Dirigida por Modelos (Ilustración 36). Este era una buena oportunidad de publicar contenido referente al TFM y realizar la propuesta previa pensada. La elección de publicar en este libro fue propuesta por parte de mi directora y una de las editoras del libro, Begoña Cristina Pelayo García-Bustelo, y de mi Codirector, Jordán Pascual Espada. La idea inicial fue que realizase la propuesta de mi TFM y lo enviase. Con ello, tendría la oportunidad de: si los revisores lo consideraban oportuno, conseguir una primera publicación; practicar la redacción de un artículo

científico; recibir críticas acerca de mí propuesta; recibir críticas acerca de la forma en que hice al artículo.



**Ilustración 36 Advances and Applications in Model-Driven Engineering**

Este capítulo presenta todas las intenciones que se realizarían en el TFM y propone un primer enfoque a la arquitectura. Este capítulo fue enviado en Febrero a la editorial IGI Global y será publicado el 31 de Agosto de 2013. Será el capítulo 4 del libro con título: *Advances and Applications in Model-Driven Engineering*. DOI: 10.4018/978-1-4666-4494-6 (González García & Pascual Espada, 2013).

### 10.3 *Domain Specific Language for the development of Educative Multiplatform Videogames Case study GADE4ALL*

Este artículo fue realizado sobre el trabajo previo a la realización del Trabajo Final de Máster. Aportó los conocimientos base sobre Ingeniería Dirigida por Modelos, Lenguajes de Dominio Específico y parte de la idea de generación de código multiplataforma a partir de un editor, así, como la adquisición de experiencia acerca de cómo realizar un artículo para una revista con índice de impacto.

Este artículo se hizo en base al trabajo realizado en el proyecto de investigación “Gade4All: plataforma genérica para facilitar el desarrollo de videojuegos y software de entretenimiento multiplataforma”, con código MITC-11-TSI-090302-2011-11 y financiado por el Ministerio de Industria, Turismo y Comercio bajo el Plan Avanza2.

Para elegir la revista a la que se enviaría el artículo se hizo un análisis de las revistas existentes. En el análisis se buscó aquellas revistas en las que se consultaron artículos, revistas con artículos del grupo de investigación y otras similares en diferentes editoriales. Estas debían estar enfocadas a temas informáticos y educativos, debido a la temática del artículo realizado. De este análisis, salieron cuatro posibles revistas pertenecientes al *Journal Citation Report*:

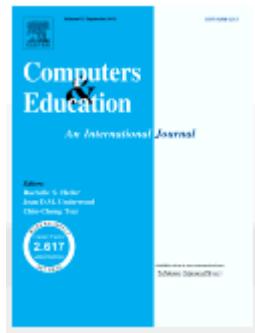


Ilustración 37 Computers & Education

**Computers & Educations** (Ilustración 37):

- <http://www.journals.elsevier.com/computers-and-education/>
- Editorial: Elsevier
- Índice de Impacto: 2.775
- Índice de Impacto (5 años): 3.305
- Categoría: COMPUTER SCIENCE, INTERDISCIPLINARY APPLICATIONS
- Ámbito: Cognición, educación, sistemas de formación, sistemas inteligentes de tutorial, hipermedia, aprendizaje de idiomas...
- Longitud de los artículos: 9-20 páginas aproximado
- Esquemas bastante libres tanto en estilo, apartados y número de referencias, así como contenido
- Entre 4 y 11 meses desde el envío hasta la publicación digital del artículo



Ilustración 38 Computer Applications in Engineering Education

**Computer Applications in engineering Education** (Ilustración 38):

- [http://onlinelibrary.wiley.com/journal/10.1002/\(ISSN\)1099-0542](http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)1099-0542)
- Editorial: Wiley Online Library
- Índice de Impacto: 0.333
- Índice de Impacto (5 años): -
- Categoría: -

Difusión de los resultados

---

- Ámbito: uses of computers, software tools in engineering education, software tools for virtual and real laboratory development, Distance learning and use of technology-based tools in classroom teaching, Use of commercial software tools in education, ...
- Longitud de los artículos: 6-15 páginas aproximado
- Contiene algunos artículos de universidades españolas. No obstante, no pertenece al índice JCR de Thomson Reuters y por ello se descartó.
- Aproximadamente 6 meses desde el envío hasta la publicación digital del artículo



Ilustración 39 Educational Technology & Society

**Educational Technology & Society** (Ilustración 39):

- <http://www.ifets.info/>
- Editorial: International Forum of Educational Technology & Society
- Índice de Impacto: 1.171
- Índice de Impacto (5 años): -
- Categoría: SOCIAL SCIENCE, EDUCATION & EDUCATIONAL RESEARCH
- Ámbito: Architectures for Educational Technology Systems, Cooperative/Collaborative Learning and Environments, Distributed Learning Environments, Educational Multimedia, Evaluation, Hypermedia Systems/Applications, Intelligent Learning/Tutoring Environments, Interactive Learning Environments, Learning by Doing, Multimedia Systems/Applications
- Longitud de los artículos: 6-20 páginas aproximado
- Esquemas bastante libres tanto en estilo, apartados, pero no pertenece a la categoría de Computer Science, si no a la de Social Science y por eso se descartó como revista.
- -



**Ilustración 40 Journal of Science Education and Technology**

**Journal of Science Education and Technology** (Ilustración 40Ilustración 37):

- <http://www.springer.com/education+%26+language/science+education/journal/10956>
- Editorial: Springer
- Índice de Impacto: 0.940
- Índice de Impacto (5 años): -
- Categoría: EDUCATION, SCIENTIFIC DISCIPLINES
- Ámbito: Teacher enhancement, computer science...
- Longitud de los artículos: 7-18 páginas aproximado
- Esquemas bastante libres tanto en estilo, apartados y número de referencias, así como contenido y temas bastante variados. No obstante, no pertenece a la categoría de Computer Science, con lo que se descartó.
- -

Se envió a la revista **Computers & Education** con índice de impacto de 2.775 y perteneciente a la editorial de Elsevier. La elección fue por ser la revista de la categoría Computer Science que mejor encajaba, tanto de estas seleccionadas como de las estudiadas previamente. Encajaba muy bien por ámbito como por artículos leídos y además, tiene un índice de impacto bueno. La fecha de envío fue el 23 de Mayo de 2013 y, actualmente, a 03 de Julio de 2013 se encuentra “*Under review*” tras haber sido asignado por el editor el día 25 de Mayo de 2013. Ref. No.: CAE-D-13-00548

## 11 REFERENCIAS

- Akyildiz, I. W. S. Y. S. E. C. (2002). A survey on sensor networks. *IEEE Communications Magazine*, (August), 102–114. Retrieved from [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1024422](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1024422)
- Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787–2805. doi:10.1016/j.comnet.2010.05.010
- Atzori, L., Iera, A., Morabito, G., & Nitti, M. (2012). The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization. *Computer Networks*, 56(16), 3594–3608. doi:10.1016/j.comnet.2012.07.010
- Blacksun Software. (2013). Mousotron. Retrieved from <http://www.blacksunsoftware.com/mousotron.html>
- Broring, A., Maue, P., Malewski, C., & Janowicz, K. (2012). Semantic mediation on the Sensor Web (pp. 2910–2913).
- Bulusu, N., Estrin, D., & Girod, L. (2001). Scalable coordination for wireless sensor networks: self-configuring localization systems. In *Proc. of the 6th International Symposium on Communication Theory and Applications (ISCTA '01)*, Ambleside, UK (pp. 1–6). Retrieved from <http://gicl.cs.drexel.edu/people/regli/Classes/CS680/Papers/Sensor%20Nets/iscta-2001.pdf>
- Deursen, A Van. (1997). Domain-specific languages versus object-oriented frameworks: A financial engineering case study. *Smalltalk and Java in Industry and Academia* ....
- Deursen, Arie Van, Klint, P., & Visser, J. (2000). Domain-specific languages: an annotated bibliography. *ACM Sigplan Notices*.
- Dijkstra, E. (1972). The humble programmer. *Communications of the ACM*, 15, 859–866.
- Eclipse. (2013a). Ecore. Retrieved from <http://wiki.eclipse.org/Ecore>
- Eclipse. (2013b). Eclipse Modeling Framework Project. Retrieved from <http://www.eclipse.org/modeling/emf/?project=emf>
- Eclipse. (2013c). Ecore Tools. Retrieved from [http://wiki.eclipse.org/Ecore\\_Tools](http://wiki.eclipse.org/Ecore_Tools)

Referencias

---

- Gama, K., Touseau, L., & Donsez, D. (2012). Combining heterogeneous service technologies for building an Internet of Things middleware. *Computer Communications*, 35(4), 405–417. doi:10.1016/j.comcom.2011.11.003
- García-Díaz, V., Fernández-Fernández, H., Palacios-González, E., G-Bustelo, B. C. P., Sanjuán-Martínez, O., & Lovelle, J. M. C. (2010). TALISMAN MDE: Mixing MDE principles. *Journal of Systems and Software*, 83(7), 1179–1191. doi:10.1016/j.jss.2010.01.010
- Georgitzikis, V., Aribopoulos, O., & Chatzigiannakis, I. (2012). Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks, 10(3), 1686–1689.
- González, E., Fernández, H., & Díaz, V. (2008). General purpose MDE tools. *IJIMAI*, 1.
- González García, C., & Pascual Espada, J. (2013). Using Model-Driven Architecture Principles to Generate Applications Based on Interconnecting Smart Objects and Sensors. In V. García Díaz, J. M. Cueva Lovelle, B. C. Pelayo García-Bustelo, & O. Sanjuán Martínez (Eds.), *Advances and Applications in Model-Driven Engineering* (pp. 73–87). IGI Global. doi:10.4018/978-1-4666-4494-6
- Groves, R. M., Fowler, F. J., Couper, M. P., Lepkowski, J. M., Singer, E., & Tourangeau, R. (2004). *Survey Methodology*. Wiley-Interscience. Retrieved from <http://books.google.es/books?id=rPJqFmQe0sYC&lpg=PR7&ots=4GjcMjKACC&dq=Fowler%2C%20M.P.%20Couper%2C%20J.M.%20Lepkowski%2C%20E.%20Singer%2C%20R.%20Tourangeau%2C%20Survey%20Methodology%2C%20Wiley-Interscience%2C%202004&lr&hl=es&pg=PP1#v=onepage&q&f=false>
- Gu, H., & Wang, D. (2009). A Content-aware Fridge Based on RFID in Smart Home for Home-Healthcare (Vol. 02, pp. 987–990).
- Guinard, D., & Trifa, V. (2009). Towards the web of things: Web mashups for embedded devices. *Workshop on Mashups, Enterprise Mashups and ....* Retrieved from <http://integrator.net/mem2009/papers/paper4.pdf>
- Guinard, D., Trifa, V., Pham, T., & Liechti, O. (2009). Towards physical mashups in the Web of Things. In *2009 Sixth International Conference on Networked Sensing Systems (INSS)* (pp. 1–4). IEEE. doi:10.1109/INSS.2009.5409925
- Han, C., Jornet, J. M., Fadel, E., & Akyildiz, I. F. (2013). A cross-layer communication module for the Internet of Things. *Computer Networks*, 57(3), 622–633. doi:10.1016/j.comnet.2012.10.003

Referencias

---

- Hao, C., Lei, X., & Yan, Z. (2012). The application and Implementation research of Smart City in China (pp. 288–292).
- Ho, D. (2013). Notepad++. Retrieved from <http://notepad-plus-plus.org/>
- Hribernik, K. A., Ghrairi, Z., Hans, C., & Thoben, K. (2011). Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino (pp. 1–9).
- IoBridge. (2013). Thingspeak. Retrieved June 23, 2013, from <http://www.thingspeak.com>
- Kasunic, M. (2005). Designing an effective survey. Retrieved from <http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA441817>
- Kent, S. (2002). Model Driven Engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 2335(2), 286–298.
- Kitchenham, B., & Pfleeger, S. L. (2002). Principles of survey research part 2: designing a survey. *IGSOFT Softw. Eng. Notes*, 27, 18–20.
- Kortuem, G., Kawsar, F., Fitton, D., & Sundramoorthy, V. (2010). Smart objects as building blocks for the Internet of things. *IEEE Internet Computing*, 14(1), 44–51. doi:10.1109/MIC.2009.143
- Lee, G. M., & Kim, J. Y. (2012). Ubiquitous networking application: Energy saving using smart objects in a home. *2012 International Conference on ICT Convergence (ICTC)*, 299–300. doi:10.1109/ICTC.2012.6386844
- Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22, 1–55.
- LogMeIn. (2013). Xively. Retrieved June 23, 2013, from <https://xively.com/>
- McFarlane, D., Sarma, S., Chirn, J. L., Wong, C. .., & Ashton, K. (2003). Auto ID systems and intelligent manufacturing control. *Engineering Applications of Artificial Intelligence*, 16(4), 365–376. doi:10.1016/S0952-1976(03)00077-0
- Meyer, G. G., Främling, K., & Holmström, J. (2009). Intelligent Products: A survey. *Computers in Industry*, 60(3), 137–148. doi:10.1016/j.compind.2008.12.005
- Miller, J., & Mukerji, J. (2003). MDA Guide Version 1.0.1. *Object Management Group*. Retrieved from <http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf>

Referencias

---

- Núñez-Valdez, E. R., Sanjuan-Martinez, O., Bustelo, C. P. G., Lovelle, J. M. C., & Infante-Hernandez, G. (2013). Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2(Regular Issue), 33–42.
- Object Management Group. (2005). MOF 2.0/XMI Mapping Specification, v2.1. Retrieved from <http://www.omg.org/cgi-bin/doc?formal/2005-09-01>
- Pintus, A., Carboni, D., & Piras, A. (2011). The anatomy of a large scale social web for internet enabled objects. *Proceedings of the Second International Workshop on Web of Things - WoT '11*, 1. doi:10.1145/1993966.1993975
- Pintus, A., Carboni, D., & Piras, A. (2012a). Paraimpu. Retrieved June 23, 2013, from <http://paraimpu.crs4.it/>
- Pintus, A., Carboni, D., & Piras, A. (2012b). Paraimpu: a platform for a social web of things. In *International World Wide Web Conference Committee (IW3C2)* (pp. 401–404). Retrieved from <http://dl.acm.org/citation.cfm?id=2188059>
- Pintus, A., Carboni, D., Piras, A., Giordano, A., & Elettrica, I. (2010). Building the Web of Things with WS-BPEL and Visual Tags Web of Things using Service-oriented Architecture standards. In *The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2010)* (pp. 357–360). Florencia, Italy.
- Piras, A., Carboni, D., Pintus, A., & Features, D. M. T. (2012). A Platform to Collect , Manage and Share Heterogeneous Sensor Data. In *Networked Sensing Systems (INSS)* (pp. 1–2). doi:10.1109/INSS.2012.6240570
- Reed, A. B., Halpern, V., & Starr, J. E. (1996). Little languages: Little maintenance? doi:10.1016/j.jvs.2012.10.105
- Roman, R., Zhou, J., & Lopez, J. (2013). On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10), 2266–2279. doi:10.1016/j.comnet.2012.12.018
- Rothensee, M. (2007). A high-fidelity simulation of the smart fridge enabling product-based services. In *3rd IET International Conference on Intelligent Environments (IE 07)* (Vol. 2007, pp. 529–532). Iee. doi:10.1049/cp:20070420

Referencias

---

- Selic, B. (2008). MDA manifestations. *The European Journal for the Informatics Professional*, ..., (June).
- Tan, L. (2010). Future internet: The Internet of Things. *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, V5–376–V5–380.  
doi:10.1109/ICACTE.2010.5579543
- Telefónica, F. (2013). *Fundación Telefónica*. Retrieved from [http://e-libros.fundacion.telefonica.com/sie12/aplicacion\\_sie/ParteA/pdf/SIE\\_2012.pdf](http://e-libros.fundacion.telefonica.com/sie12/aplicacion_sie/ParteA/pdf/SIE_2012.pdf)
- The US National Intelligence Council. (2008). *Six Technologies with Potential Impacts on US Interests out to 2025*.
- Thomson Reuters. (2013). JCR. Retrieved from <http://thomsonreuters.com/journal-citation-reports/>
- Wong, C. D. Mf. A. A. V. A. (2002). The intelligent product driven supply chain. In *Systems, Man and ...* Retrieved from [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1173319](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1173319)
- World Wide Web Consortium. (1999). XSLT. Retrieved from <http://www.w3.org/TR/xslt>
- World Wide Web Consortium. (2004). XML. eXtensible Markup Language. Retrieved from <http://www.w3.org/XML/>

## 12 ANEXO I: ACRÓNIMOS

**CIM** Computational-Independent Model

**CORBA** Common Object Request Broker Architecture

**CWN** Common Warehouse Metamodel

**DSL** Domain Specific Language

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**IoT** Internet of Things

**IoTU** Usuarios interesados en Internet of Things

**ISM** Implementation-Specific Model

**JCR** Journal Citation Report

**JSON** JavaScript Object Notation

**MDE** Model-driven engineering

**MOF** Meta-Object Facility

**NFC** Near field communication

**NoSQL** No Structured Query Language

**OMG** Object Manager Group

**PIM** Platform-Independent Model

**PSM** Platform-Specific Model

**REST** Representational State Transfer

**RFID** Radio Frequency IDentification

**SDev** Software Developers

**SIoT** Social Internet of Things

**SOA** Service-oriented architecture

**SQL** Structured Query Language

**UML** Unified Modeling Language

**WSAN** Wireless sensor and actor networks

Anexo

---

**WWW** World Wide Web

**XMI** XML Metadata Interchange

**XML** eXtensible Markup Language

## 13 ANEXO II: GLOSARIO

**Android:** Sistema operativo basado en Linux utilizado para móviles.

**Arduino:** Plataforma libre que consiste en una placa, un micro-controlador y un entorno de desarrollo que facilita el uso de electrónica.

**CORBA:** Estándar definido por Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras, puedan trabajar juntos; es decir, facilita el desarrollo de aplicaciones distribuidas en entornos heterogéneos.

**Common Warehouse Metamodel:** Especificación para modelar metadatos relacionales, no relacionales y objetos. Esta especificación es del Object Management Group.

**Eclipse:** Programa informático usado como entorno de desarrollo integrado.

**Framework:** Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

**HTML:** Lenguaje de marcado predominante para la creación de sitios web. Es un estándar..

**HTML5:** Versión 5 del estándar HTML.

**HTTP:** Protocolo de transferencia de hipertexto usando en las transacciones de la World Wide Web (WWW).

**Internet of Things:** Red inteligente que contiene objetos heterogéneos y ubicuos interactuando entre ellos.

**Java:** Lenguaje de programación de alto nivel orientado a objetos creado por Sun Microsystems.

**Javascript:** Lenguaje de programación interpretado y basado en prototipos. Es un dialecto del estándar ECMAScript.

**Journal Citation Report:** Publicación anual que realiza el Institute of Scientific Information, miembro de la empresa Thomson Scientific. Proporciona información sobre revistas científicas del campo de las ciencias aplicadas y sociales. Originalmente era parte del Science Citation Index. Actualmente está realizado a partir de los datos que este contiene

**JSON:** Formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

**Lenguaje de Dominio Específico:** Especificación de un lenguaje dedicado a resolver un problema en particular, representar un problema específico y proveer una técnica para solucionar una situación particular.

**Micro-controlador:** Circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria.

**Modelo:** Según la Real Academia Española, la palabra modelo tiene varios significados:

1. Arquetipo o punto de referencia para imitarlo o reproducirlo.
2. En las obras de ingenio y en las acciones morales, ejemplar que por su perfección se debe seguir e imitar.
3. Representación en pequeño de alguna cosa.
4. Esquema teórico, generalmente en forma matemática, de un sistema o de una realidad compleja, como la evolución económica de un país, que se elabora para facilitar su comprensión y el estudio de su comportamiento.

En nuestro caso, un modelo es una abstracción de un sistema del mundo real y que describe aquellos aspectos del sistema que son relevantes desde su punto de vista con un apropiado nivel de detalle.

**MOF:** Estándar del grupo OMG para la Ingeniería Dirigida por Modelos. Es una arquitectura de metamodelado cerrada. Con ella se define UML.

**Near Field Communications:** Tecnología de comunicación inalámbrica, de corto alcance y alta frecuencia que permite el intercambio de datos entre dispositivos. Los estándares de NFC cubren protocolos de comunicación y formatos de intercambio de datos, y están basados en ISO 14443 (RFID).

**NoSQL:** Sistemas de gestión de bases de datos que difieren del modelo clásico del sistema de gestión de bases de datos relacionales (RDBMS) en aspectos importantes, el más destacado que no usan SQL como el principal lenguaje de consultas. Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN, ni garantizan completamente ACID (atomicidad, coherencia, aislamiento y durabilidad), y habitualmente escalan bien horizontalmente.

**NotePad++:** Editor de texto y de código fuente libre con soporte para varios lenguajes de programación. De soporte nativo a Microsoft Windows. Similar al Bloc de Notas pero que incluye opciones más avanzadas que pueden ser útiles para usuarios avanzados como desarrolladores y programadores.

**Object Management Group:** Consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones

para las mismas. El grupo está formado por diversas compañías y organizaciones con distintos privilegios dentro de la misma

**POST:** Uno de los muchos métodos de solicitud soportadas por el protocolo HTTP utilizado por la World Wide Web. El método de la petición POST está diseñado para solicitar que un servidor web acepte los datos adjuntos en el cuerpo del mensaje de solicitud para el almacenamiento. A menudo se utiliza al cargar un archivo o enviar un formulario web completa.

**REST:** Técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

**RESTful:** Sistemas que siguen los principios REST.

**RFID:** Sistema de almacenamiento y recuperación de datos remotos que usan dispositivos denominados etiquetas, tarjetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (Automatic IDentification, o identificación automática). Son unos dispositivos pequeños, similares a una pegatina, que pueden ser adheridas o incorporadas a un producto, un animal o una persona

**Ruby on Rails:** Framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby, siguiendo el paradigma de la arquitectura Modelo Vista Controlador (MVC). Trata de combinar la simplicidad con la posibilidad de desarrollar aplicaciones del mundo real escribiendo menos código que con otros frameworks y con un mínimo de configuración. El lenguaje de programación Ruby permite la metaprogramación, de la cual Rails hace uso, lo que resulta en una sintaxis que muchos de sus usuarios encuentran muy legible. Rails se distribuye a través de RubyGems, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby.

**Sensor:** Dispositivo capaz de detectar magnitudes físicas o químicas, llamadas variables de instrumentación, y transformarlas en variables eléctricas. Las variables de instrumentación pueden ser por ejemplo: temperatura, intensidad lumínica, distancia, aceleración, inclinación, desplazamiento, presión, fuerza, torsión, humedad, movimiento, pH, etc. Una magnitud eléctrica puede ser una resistencia eléctrica (como en una RTD), una capacidad eléctrica (como en un sensor de humedad), una Tensión eléctrica (como en un termopar), una corriente eléctrica (como en un fototransistor), etc.

**SQL:** Lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar de forma sencilla información de interés de bases de datos, así como hacer cambios en ella.

**Smart City:** Aquellas ciudades que aplican las Tecnologías de la Información y las Comunicaciones en la misma ciudad con el objetivo de proveerla de una infraestructura que, en cierto grado, garantice un

desarrollo sostenible, un incremento de la calidad de vida de sus ciudadanos, una mayor eficiencia de sus recursos (de tipo humano como energético) y una mejor participación ciudadana.

**Smart Earth:** Este concepto trata sobre extender Internet of Things por el medio ambiente y las instalaciones creadas por el ser humano (presas, ferrocarriles, túneles, edificios, carreteras,...), para así, extender la información que se reciba del mundo físico.

**Smart Home:** Engloba la automatización de la actividad en el hogar y las tareas domésticas. La domótica puede incluir un control centralizado de la iluminación, HVAC (calefacción, ventilación y aire acondicionado), los electrodomésticos, y otros sistemas, para proporcionar mayor comodidad, el confort, la eficiencia energética y la seguridad.

**Smart Object:** Elemento físico, con diversas propiedades, identificable a lo largo de su vida útil, que interactúa con el entorno y otros objetos y que puede actuar de manera inteligente según unas determinadas situaciones, mediante una conducta autónoma.

**Smartphone:** Teléfono móvil construido sobre una plataforma informática móvil, con una mayor capacidad de almacenar datos y realizar actividades semejantes a una mini computadora y conectividad que un teléfono móvil convencional. El término «inteligente» hace referencia a la capacidad de usarse como un ordenador de bolsillo, llegando incluso a remplazar a un ordenador personal en algunos casos.

**Smart TV:** Integración de Internet y de las características Web 2.0 a la televisión digital, así como la convergencia tecnológica entre los ordenadores y estos televisores. Estos dispositivos se centran en los medios interactivos en línea, en la televisión por Internet, y en otros servicios como el vídeo a la carta.

**Tablet:** Computadora portátil de mayor tamaño que un teléfono inteligente o una PDA, integrado en una pantalla táctil (sencilla o multi-táctil) con la que se interactúa primariamente con los dedos o una pluma Stylus (pasiva o activa), sin necesidad de teclado físico ni ratón. Estos últimos se ven reemplazados por un teclado virtual y, en determinados modelos, por una mini-TrackBall integrada en uno de los bordes de la pantalla.

**UML:** Lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.

**WSAN:** Grupo de sensores y actores unidos por medios inalámbrico para llevar a cabo detección distribuida y tareas de accionamiento. En este tipo de redes, los sensores recogen información sobre el mundo físico, mientras que los actores toman decisiones y realizan acciones apropiadas sobre el medio ambiente, lo que permite a distancia, la interacción automática con el medio ambiente.

**World Wide Web:** Sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza sitios web

Anexo

---

compuestos de páginas web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navega a través de esas páginas usando hiperenlaces.

**XMI:** Especificación para el Intercambio de Diagramas escrita para proveer de una manera de compartir modelos UML entre diferentes herramientas de modelado.

**XML:** Lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes.

## 14 ANEXO III: ÍNDICE ALFABÉTICO

	15	H
.NET	- 14 -	HTML5
<b>A</b>		HTTP
Actuators Networks	- 10 -	<b>I</b>
Android	- 18 -, - 32 -, - 33 -	Índice de Impacto
Arduino	- 1 -, - 2 -, - 5 -, - 11 -, - 17 -, - 18 -, - 19 -, - 27 -, - 28 -, - 32 -, - 33 -, - 36 -	- 62 -, - 63 -, - 64 -, - 65 -, - 67 -, - 68 -, - 69 -
arquitectura dirigida por modelos	- 13 -	Ingeniería Dirigida por Modelos
<b>C</b>		Internet of Things II, V, - 1 -, - 2 -, - 4 -, - 5 -, - 7 -, - 9 -, - 21 -, - 37 -, - 46 -, - 47 -, - 54 -, - 56 -, - 60 -, - 62 -, - 63 -, - 64 -, - 85 -
CIM	- 13 -, - 14 -	IoBridge
CORBA	- 14 -	IoT- 1 -, - 4 -, - 7 -, - 9 -, - 17 -, - 18 -, - 19 -, - 22 -, - 36 -, - 60 -, - 61 -
COSM	- 18 -	IOT
CWN	- 14 -	IoTU- 36 -, - 46 -, - 47 -, - 54 -, - 55 -, - 56 -, - 57 -
<b>D</b>		ISM
Deepwater Horizon	- 2 -, - 8 -	<b>J</b>
diagrama de cajas y bigotes	- 38 -, - 39 -	Java- 12 -, - 14 -, - 18 -, - 25 -, - 27 -, - 29 -, - 30 -, - 77 -
DSL IV, - 3 -, - 6 -, - 7 -, - 9 -, - 12 -, - 15 -, - 21 -, - 24 -, - 25 -, - 27 -, - 29 -, - 36 -, - 43 -, - 59 -, - 60 -, - 62 -		Javascript
<b>E</b>		Journal Citation Report
Eclipse	- 12 -, - 22 -	JSON
Editor Web Gráfico	- 26 -	<b>L</b>
Editor Web Textual	- 25 -	Lenguaje de Dominio Específico II, - 3 -, - 6 -, - 7 -, - 9 -, - 12 -, - 13 -, - 15 -, - 21 -, - 22 -, - 25 -, - 27 -, - 59 -, - 60 -, - 62 -
Extensible Stylesheet Language Transformations	- 16 -	<b>M</b>
<b>F</b>		MDA
Foursquare	- 17 -	MDE

Anexo

---

metamodelo	- 12 -, - 22 -, - 23 -, - 24 -	<b>S</b>
MIDGAR 1, - 21 -, - 22 -, - 25 -, - 26 -, - 31 -, - 34 -, - 61 -, - 62 -, - 85 -		SDev - 36 -, - 46 -, - 47 -, - 52 -
<b>Modelo</b>	- 22 -, - 78 -, - 79 -	SenseWeb - 17 -
MOF	- 14 -	Sensorpedia - 17 -
<i>Mousotron</i>	- 35 -	Sintaxis abstracta - 13 -, - 24 -
		Sintaxis concreta - 24 -
		SIoT - 23 -
		Smart Cities - 1 -
Near Field Communications	- 9 -	Smart City - 5 -
Netduino	- 19 -	Smart Earth - 1 -
Nexus 4	- 33 -, - 36 -	Smart Home - 1 -, - 31 -
NFC	- 9 -, - 61 -	Smart Object - 4 -, - 5 -, - 10 -, - 11 -, - 32 -
<b>NoSQL</b>	- 61 -, - 75 -	Smart Objects - 2 -, - 3 -, - 4 -, - 9 -, - 10 -, - 11 -, - 17 -, - 18 -, - 22 -, - 31 -, - 37 -, - 63 -, - 64 -
Notepad++	- 34 -, - 36 -, - 39 -, - 40 -, - 43 -, - 44 -, - 45 -, - 46 -	Smart TVs - 10 -
		Smartphones - 1 -, - 2 -, - 10 -, - 61 -
		SOA - 5 -
<b>O</b>		<b>SQL</b> - 15 -, - 61 -, - 75 -
		<b>T</b>
Pachube	- 17 -, - 18 -	Tablets - 1 -, - 10 -
Paraimpu	- 17 -	ThingSpeak - 17 -, - 19 -, - 20 -
PIM	- 14 -	Twitter - 17 -, - 61 -
POST	- 20 -	
PSM	- 14 -	<b>U</b>
		UML - 14 -
<b>Q</b>		<b>W</b>
QuadraSpace	- 17 -	WSAN - 10 -
<b>R</b>		<b>X</b>
REST	- 18 -, - 19 -, - 20 -, - 30 -	Xively - 17 -, - 18 -, - 19 -
RESTFul	- 79 -	XMI - 14 -
RESTFull	- 5 -	XML - 14 -, - 17 -, - 19 -, - 21 -, - 29 -, - 34 -, - 39 -, - 76 -, - 77 -, - 81 -
RFID	- 1 -, - 2 -, - 9 -, - 61 -	
Ruby on Rails	- 33 -, - 61 -	XSLT - 16 -

## 16 ANEXO IV: CONTENIDO DEL DVD-ROM

En este anexo se describe el contenido que va dentro del DVD-Rom.

### 16.1 *Prototipos*

En esta carpeta se encuentran todos los prototipos desarrolladoras para esta investigación.

- ArduinoProgram (C)
  - Código creado para la ejecución de órdenes en el Arduino y que se introduce en la memoria de este.
- MidgarAppAndroid (Java-Android)
  - Aplicación para los dispositivos con sistemas operativos Android. Esta les permite conectarse con el servidor MIDGAR para enviar datos y recibir datos, así, como ejecutar las acciones en base a ellos.
- MidgarAppArduino (Java)
  - Aplicación para conectar los dispositivos Arduino a un ordenador por cable USB. Esta les permite conectarse con el servidor Midgar para enviar datos y recibir datos, así, como ejecutar las acciones en base a ellos.
- MidgarAppGenerator (Java)
  - Aplicación generadora de las aplicaciones que interconectan objetos en base a lo definido por el usuario con el DSL.
- MidgarAppServer (Java)
  - Aplicación que se utiliza como plantilla y en la que se sustituye lo definido por el usuario en el DSL. Esta plantilla es la base de las aplicaciones que interconectan los objetos corren en el servidor.
- MidgarServer (Ruby on Rails)
  - Red IoT MIDGAR.

### 16.2 *Artículos*

Todos se componen de una carpeta subida en la que está toda la información que se envió cuando se subió el artículo, así como la recibida de las editoriales. En la carpeta raíz se encuentra la versión íntegra del artículo.

## 17 ANEXO V: ARTÍCULOS

- 17.1 *MIDGAR: Generation of heterogeneous objects interconnecting applications. A Domain Specific Language proposal for Internet of Things scenarios*

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

Cristian González García<sup>a\*</sup>, Cristina Pelayo García-Bustelo<sup>a</sup>, Jordán Pascual Espada<sup>a</sup>

<sup>a</sup> University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain. Tel: +34985103397

<sup>a</sup> gonzalezgarciacristian@hotmail.com, crispelayo@uniovi.es, pascualjordan@uniovi.es

\* Corresponding author.

### ABSTRACT

Smart Objects and Internet of Things. Two ideas that describe the future, the interconnection of objects in order to make them intelligent or expand their intelligence if they already had it. All this is done by a network that connects all the objects in the world. A network where most of the data traffic comes from objects instead of persons. Cities, houses, cars or 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 that make possible for inexperienced people to interconnect them. For that purpose, we present two possible solutions: a Domain Specific Language capable of abstracting the problem of the generation of applications and a graphic editor that simplifies the creation of that DSL.

**Keywords:** Internet of Things; ubiquitous computing; sensor network;

## 1 INTRODUCTION

Internet of Things (IoT) opens the door to many possibilities, like the field of environmental intelligence [14], better known as Smart Earth [18], the Smart Cities [18] or the Smart Homes [14,17,19]. This is due to a recent rise in the use of new technologies, like computers, the Internet, cloud computing, Smartphones [40], tablets, micro-controllers [11,34] and intelligent tags [23,39]. Also, the prices of the sensors are now much lower than before and the wireless connections have been quite improved [1]. All this set of objects and new technologies explains that the Internet of Things is gaining a lot of importance in the field of modern telecommunication [2].

IoT allows us to have sensors and objects dispersed so they can send us information from any place 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, or inside a nuclear plant, in a sealed room full of chemical substances, or maybe among the enemy lines in a battlefield or even in places where they can't be reached after being set [1,5]. With this we can establish an interconnection between those heterogeneous objects as long as they have access to the Internet [24]. These objects can be sensors dispersed through the terrain [1], a computer[36], a net of sensors [1], an Arduino micro-controller, a Smart TV, a Smartphone, a Radio Frequency ID (RFID) intelligent tag, a freezer... Internet of Things is the interconnection of heterogeneous objects through the Internet [3,19,23,24]. This will lead to a future in where it won't be used just to communicate people, but also to communicate people with machines and even machines with machines (M2M) [39]. Thanks to this, Internet of Things will become a great impact in our daily routines, both for personal and commercial use [2], as it can improve our quality of life and the production of an enterprise. All through the interconnection of objects and all the possibilities this offers.

It must be highlighted that Internet of Things is gaining so much importance that even the United States National Intelligence Council sees it as one of the six technologies that will have more weight in the interests of the United States from here to 2025 [41]. On the other hand, back in 2005, at Tunisia, the ONU predicted a new era of ubiquity where the internet traffic will be small in comparison to the traffic generated by objects from our daily routines [2].

That's why Smart Objects are also important: these devices can interact with others thanks to certain actions or events [19]. An example of this can be found in Smartphones, micro-controllers like Arduino [11,19,34] or any other object capable of managing information [28]. 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 this objects.

For instance, [14,37] present a freezer that analyzes the habits of alimentation and can be very useful for people with an illness or even to improve the quality of life of a family. [19] investigated an intelligent system to give a certain amount of intelligence to a house and control a forklift thanks to certain conditions that activated its different

Anexo

---

sensors. This can also help ecology and economy, because it can save energy in a campus by using RFID intelligent tags, as [39] proposed, but this could also be applied to a house [24].

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 [4]. For instance, in the disaster of the Deepwater Horizon in 2010 at Golfo de México it could have been used to control the spill of oil, analyze the amount of salt in the water and the status of different parts to prevent bigger problems [4], or even (before the accident) help to prevent many disasters by monitoring different structures and its condition.

Because of this, the purpose of this investigation is to design a Domain Specific Language (DSL) and a graphic editor to implement it. This editor will allow us to create applications to interconnect heterogeneous objects: sensors, Smart Objects, etc. We intend to make possible for anyone to interconnect many objects without programming knowledge. Thanks to this DSL we offer the user an abstraction of the base programming language, so he will just have to know the domain of the problem and its properties, because the editor will make the writing process easier for him. Also, this 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 and solving the interconnection problem.

The rest of this paper is structured this way: In section 2, there is an introduction to the currently existing problems in IoT. In section 3 we explain the contribution that is being made through this research, towards solving the aforementioned problems. In section 4 we talk about Internet of Things, Smart Objects, Model Driven Engineering and the DSL and there is also a research on the existing IoT platforms. Section 5 shows the case study of this research in detail: the Midgar IoT platform. In section 6 we have the evaluation and discussion of the data obtained from the surveys done on the participants. Section 7 covers the conclusions of this paper. In Section 8 we can find the many possible options for future work that can be done from the results of this research

## 2 PROBLEMS

With Internet of Things and the Smart Objects we can create a Smart network that controls almost anything. However, there is a big common problem. The heterogeneity of the problems doesn't allow them to interact with each other [9]. To connect them, we need to develop a common interface, like an application hosted in both of them that consults a server. We can see an example of this in the existing multiplatform programs, from messaging services like WhatsApp, Skype or Line, to multiplayer videogames like Triviados y Apalabradados. However, these applications can only connect certain predefined Smart Objects and can only be developed by people with knowledge in informatics. IoT, however, offers a great support for heterogeneous objects, no matter the operative

Anexo

---

system, the platform or their functionalities. Also, it doesn't matter if the objects are intelligent or not or if they are domestic objects or sensors scattered throughout the world.

People without programming knowledge that wants to connect those objects to take advantage of the technology and help creating a better environment. These people should only have to know about the domain of the problem instead of having to learn how to program. Because of this, the solution to this problem is to create a domain specific language to reach that abstraction layer and encapsulate the domain of the problem in it [7].

## 2.1 *Heterogeneity of the objects*

The main problems of the Internet of Things and the 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 is the different implementations that must be done for the different operative systems. From the software's point of view, what can differ in each case is the type of message that is being sent. That's why the messages sent between Smart Objects created by an enterprise can be very different from the ones sent by the Smart Objects of another. This makes impossible for Smart Objects of different enterprises to interact, because there is no understanding between them [9]. The solution for these problems would be using a network which could interconnect the different devices [39]. 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. Because of this, the main problem is the heterogeneity, the dynamism and evolution of its content [9]. 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.

These 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 so little in common with the rest. There is not a standardization that encompasses all of these objects in every aspect (protocols, available services, types of data, sending of messages...). 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 [15]. As an exception, we should mention Bluetooth and infrared sensors. This difference between objects is what we call object heterogeneity. That's why we must provide an adequate architecture which 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 [11,19]. In this case, the heterogeneity of the 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 are the Smart Cities. Has [18] says, the first problem that must be solved is the recollection of data, the access and the transmission. This is because Internet of

Things needs to recognize the 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 [1]. That implies multiple solutions but none of them can become a standard [15].

In this case, the solution is an architecture that supports the sending of the messages coming from the different types of devices and it's able to respond to each one. For that, we will create a service-oriented architecture (SOA), as proposed [2,9,30,33] and a RESTFull application that can be used by any object, as proposed in [16].

## 2.2 *Development of applications to interconnect objects for unexperienced users*

The development of applications requires people with experience in software development that have a wide knowledge of the domain of sensors and objects to interconnect, as well as knowledge of the different programming languages that are needed to use in each platform. The main problem is to make possible for people without informatics knowledge to develop these applications in a quick and simple way [15]. The solution is to offer an application to help them with that 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's 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

Objects interacting between them and sending messages to each other. 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. This is Internet of Things and many of these objects are 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, so they can work together within the same network, under the same protocol or standard (Guinard & Trifa, 2009). Second, not everyone is able to program. Both problems could be solved by themselves. The first, establishing a common standard for all the creators of objects. The second, teaching everyone how to program. However, none of these solutions is viable.

Because of this, a possible solution would be to make possible for 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 encompasses this domain.

### 3.1 *Internet of Things*

Nowadays, most of the interactions performed through the Internet are of the human-human type. 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 [39]. 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's Internet of Things [2,9,19,23,24,39]. 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 [2,9,23]. This is because, with the use of RFID, we can assign a single identifier to an object [2]. This way, we can locate a certain object for a certain task. However, as [39] explains, for the Internet of Things to exist, three steps are required: integrated intelligence, connectivity and interaction. That's why the base for Internet of Things are the sensors and the RFID cards [2,39]. However, these are not the only ones. According to [2], 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 [9], so, through radiofrequency, we can know the properties of this element and monitor it [23]. 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 this, we need an intelligent infrastructure capable of connecting the different objects and granting them enough intelligence to interact with each other or communicating them the decision they have to make [39]. This last thing would only be necessary if the Smart Objects don't have intelligence of their own or if they need it in a certain moment because of an external decision.

### 3.2 *Smart Objects*

As [2,19,27,42] 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 [19,28], 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.

This way, we can classify a 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 [21], 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, problems that were already present in the 60s and are still present, as it can be seen in [8,12]. The solution to this problems can be obtained through the automation or semi-automation of processes, something in which MDE is quite popular [10]. With this we manage to reduce the complexity of the design and the implementation, which helps to obtain a much more reliable software and with more sophisticated functionalities [38]. 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. This 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 [29]. 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 [38].

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 [7,29]. 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 [6,7]. 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 [7,35]. However, the Domain Specific Language are very important in Model Driven Engineering [29].

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 provider: Xively. Others are aimed at investigations, like Paraimpu and QuadraSpace. 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

Anexo

---

invitations or previously approved accounts. Some examples of this are ThingSpeak, Sensorpedia and SenseWeb. In this section we will describe Paraímpu, Xively and ThingSpeak. The choice is clear: these are the ones that offer more data for the research on how they work. The other networks weren't accessible due to their beta state or because of the little information they provided. The three 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 Paraímpu**

It's 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's 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().toString())%> and we have <% NameOfTheSensor.value%> <% NameofTheSensor .unit%>*". It allows users to configure over the level of privacy when sharing the sensors [31,32,30,34].

Besides, as it can be seen in the examples they provide us with, 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 need 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 the objects.

With its DSL and its graphic editor, Midgar offers users a higher level of abstraction through the use of tags, thus preventing them from having to program. Besides, it offers more than just the condition, 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 putting 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 the 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 service and communicate the action to it by using a REST method [26].

Xively offers many privacy settings and access to REST through other service, and it also make easier to connect objects through its trigger creator. These are focused in users without knowledge in programming. However, Xively doesn't allow the performing of one single condition and significantly reduces the possibilities, while Midgar makes possible to create an application that can connect an infinite number of objects and triggers thanks to the DSL.

### 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...). After that, it allows us to create a new channel. Channels are used to upload the data of 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 if the data is accessible to everyone or only the user [20].

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.

## 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, without needing programming knowledge. To interconnect them, the user can use the graphic editor to write the application by using the DSL created for this case. He won't be forced to user the given editor if he doesn't

Anexo

---

wants to. However, as it will be explained in the **Error! No se encuentra el origen de la referencia.**, this will help him in an optimal way while creating the DSL. Thanks to this DSL, a user can describe all the flow that he wants to be performed by the application. With this application, 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. This way, 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.

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

#### 4.2 *Proposed architecture*

The system's architecture can be divided in four layers, as it can be seen in Figure 1. Each one is a process within the global set of the infrastructure. So, the first layer encompasses the user's process. After this, one user must define the purpose of his application by using the web editor. Once it is finished, the DSL that contains its information is generated in XML format. The information is moved to the second layer: the service generation layer. This layer processes the information, generates a Java application from it, compiles it and then executes it in the server. This last step is part of the third layer, the server layer. This way, the application created by the user keeps working in the server, performing the task defined by the user. As long as it is functioning, it will directly communicate with the server and the data base. Also, 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. The Smart Objects are found in the last layer. These implement the message interface to keep a permanent, bidirectional connection with the server.

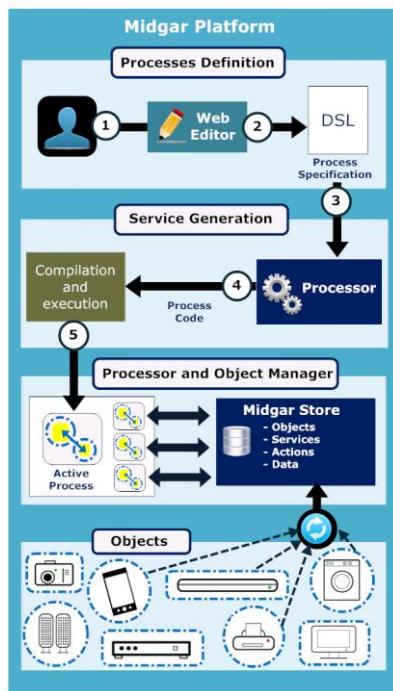


Figure 1 Architecture of the Midgar Platform

#### 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 begin explaining how it Works. After that, we will 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 will be the next one, in which we will explain how the DSL is processed 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 show how the connection of objects with the platform works.

##### 4.3.1 *Graphic Web Editor*

The editor created for the Midgar Project is a graphic editor (Figure 2). This way, if the user wishes to generate an application, all he has to do is selecting the box that corresponds with the flow he wants to add to the program and link it to the rest by using connectors. The editor can be divided in five zones.

Anexo

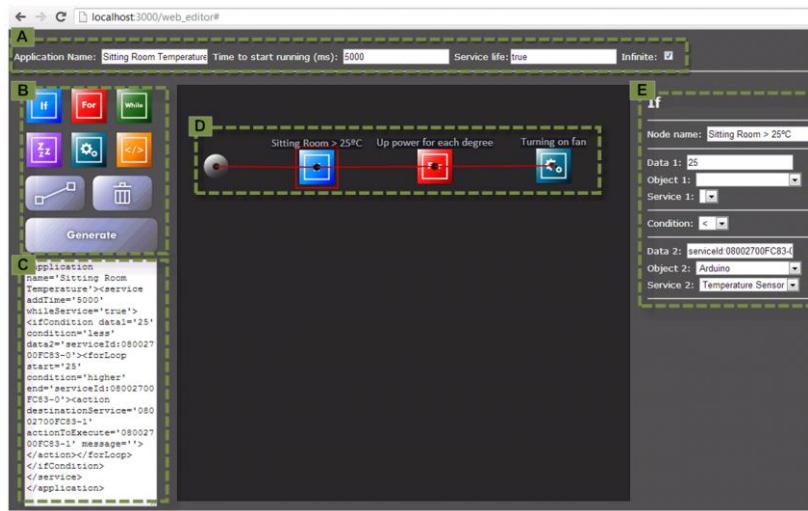


Figure 2 Editor for the generation of object-interconnection applications of the Midgar platform

The first one is shown in Figure 2A. This zone contains the data of the application we want to create: name, the time that passes 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 zone (Figure 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”, which includes a condition for the flow; the “For”, which allows the generation of loops that will be executed as long as a certain condition is met; the “While”, a loop that is very similar to the “For” but gives support to the time-based conditions; the “Sleep”, which makes possible for the application to sleep during a certain amount of time and the “Action”, the final and most important element, because its inclusion makes possible to send and order (along with a message) to an object so it executes it.

Insertion of code, in this case it's possible to add code in the application so we can include any kind of flow supported by the base language (Java) and perform object interaction requests. By using this button we create the connection between the elements added to the application and thus we establish the flow and the order that the application must follow. Next to it is the “Erase” button, which allows us to eliminate any element of the flow. Finally, the “Generate” button allows us to generate the application and visualize the DSL in the editor.

Figure 2C contains the visualization of the DSL corresponding to application created by the user with the editor. In the central area is the work zone. Here is where the application is created and the flow connection is performed. The application's flow always starts from the gray ball, as shown in Figure 2D. The application shown in the

Anexo

---

example checks the value of 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, Figure 2E shows the questionnaire zone. Here we can visualize the questionnaire belonging to the selected node in the work zone. In this case, the select node is an “If”.

#### 4.3.2 DSL

The Domain Specific Language created for the generation of applications that interconnect devices has a total of six nodes. Four of them correspond to programming language structures, and the remaining two are one that corresponds to the insertion of direct code and language and other that corresponds to the action that has to be executed. The first four are the conditional, two loops and the sleep. 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’s execution time by adding pauses or waits. To create the program’s flow through the use of the DSL, the user must nest one inside another if he wants them to be executed when the father’s 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 milliseconds) 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 or 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’s 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'>
```

Anexo

---

```
<action destinationService='c3b9f28c24f2be8b-0' actionToExecute='2' message='This action is a
popup'></action>

</ifCondition>
```

Another example would be to use 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 checked the temperature of a sensor placed in an Arduino micro-controller. When this one surpassed the 25°C, it sent a vibration (2500 milliseconds per each extra grade) to the user's mobile phone. This service could be useful to control places in which the temperature raises a lot and inform the controller in a not very intrusive way. Another use for this could be 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 it 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 it could be seen in the previous example, the two actions and the sleep are executed within the body of the loop, which is it only executed when a certain condition is met. That's 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 informatics 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 this 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 is the vibration and sends a parameter of 4000, equivalent to 4 seconds 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 Figure 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 in XML format with 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, 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 do different things 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 compiles it and executes it in the server.

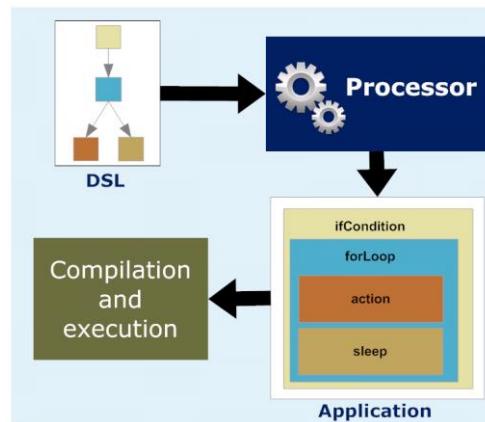


Figure 3 Internal working of the Service Generation

#### 4.3.4 Processor and Object Manager

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

Anexo

---

**Process server:** In this layer we can find the applications developed by the users. 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.** This one 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 [2,9,30,33], as well as a REST architecture, as [16] suggest, for its many benefits and its easy use. This way, it becomes accessible to any object that implements the same message language tan the Midgar platform, and by using the REST, the 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, it generates the response with all the pending messages for the Smart Object and sends them to it in reply to the requests. If there are not, it sends a message to confirm the correct reception of the information.

#### **4.3.5 *Objects***

The 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 that makes necessary for the object to present its actions and services. The first thing for an object to do is registering itself in the platform. By doing so, the actions and services of the object are registered in Midgar's data storage so the users can use the services of the registered objects in the specification of the object-coordination process.

The 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 and the switchboard of a Smart Home could send data about each room's temperature and humidity, or even communicate if the doors and windows are open or closed.

The 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, while a Smart Home could open and close doors and Windows or turn the heater and the humidifier on or off. Once it's registered, the object can start sending data, including data of every service in each sending. After that, 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.

##### **4.3.5.1 *Android***

Anexo

---

The native application developed for Android mobile phones offers support starting from the 2.1 version, which corresponds with API7. This shows a list with the sensors of the mobile phone we are using and the values they capture in real time. The project makes possible to modify the sensors we want to send in an easy way, as well as writing the registration message of the mobile phone's services and actions. After that, the application is deployed in the mobile device, we select the registration button and run the application to start the sending of 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 with the graphic editor.

#### 4.3.5.2 Arduino

The Arduino micro-controller (Figure 4) was connected to the PC via USB. For the sending and reception of messages we used a Java application that acts as intermediary for the Arduino. This way, we facilitate its use, because the user doesn't have to use the libraries in C or program at all. Besides, it makes 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 had to install the sensors in the micro-controller, configure them in C (by using the Arduino's own IDE) and then load the program. After that, we execute the Java application and it's ready to be used. First we register it and then the transmission of data starts automatically.

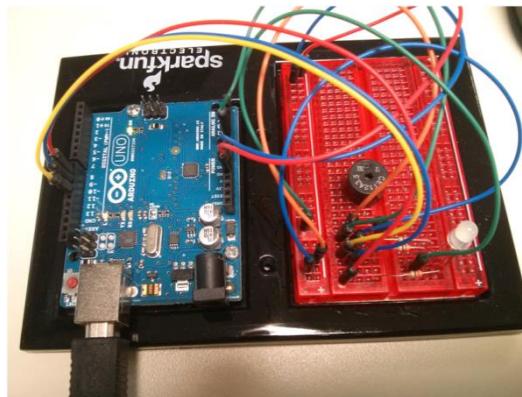


Figure 4 Arduino picture during a testing

#### 4.4 *Software used*

For this research, we used many different types of hardware and software:

- The server of the investigation was created with the Ruby on Rails framework
- The editor was developed by using HTML5 and Javascript.
- To create the generation of applications and the connection application for the Arduino, we used Java6. This way, we gained the multiplatform flexibility that could have been necessary in both cases

For the tests, we used the following equipment:

Anexo

---

- A server hosted in a dedicated computer with a 64 Bit Windows Server 2008R2 operative system, a 3100MHz Intel Core i3-2100 processor and 4GB of RAM memory.
- Webrick web server.
- SQLite database.
- Four Smartphone: 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 micro-controller.

## 5 EVALUATION AND DISCUSSION

To be able to evaluate the easiness to create applications that interconnect heterogeneous objects, we choose to do a survey on the participants of the tests. This method is used in the software engineering field to provide information that effectively supports the taking of decisions (Kasunic, 2005). 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 if the work we performed was successful or not.

In this section we present the performed experiment in detail and show the obtained results. In the first part of the evaluation we will describe the methodology used to perform the tests. After that, we will show and discuss the obtained results.

### 5.1 *Methodology*

As a measurement method for the survey, we used the Likert Scale [25]. We choose this one because it's the most used in the design of scales. We choose 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 choose a size that represented the population and could offer significant results for the study [22]. For the user profiles we choose software developers (SDev) and people interested in IoT (IoTU). The 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, software developer or not.

To perform the population sampling we used the snow ball method [13]. 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, while 9 were people interested in IoT. 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.

For them to be able to perform the tests, they first had to use the graphic editor. During all the process they were individually taught how to use the editor. First, they were explained about the scenario of the project, the current problems and the research. Later, they were taught which applications they should use in a real case. This way, the

Anexo

---

users had a better understanding. They had to check the temperature of the Arduino micro-controller. When it surpassed the 25°C it should send a vibration of 2500 milliseconds to the Nexus 4, as well as a notification to the Motorola.

Once the users understood this, they were briefly shown the graphic editor: placement of the buttons, meaning, distribution, use, etc. After this procedure, they created the application with the graphic editor and after finishing, they moved to another computer and started to fill their surveys, anonymously and without help (Tabla 2).

<b>Question</b>	<b>Description</b>
<b>Q1</b>	This tool provides useful assistance like an improvement editing for the creation of applications which interconnect objects.
<b>Q2</b>	This editor offers an effective way to developing the indicated task.
<b>Q3</b>	The way to create applications using the editor is understandable.
<b>Q4</b>	The use of the editor reduces complexity of developing these type applications.
<b>Q5</b>	The editing offers enough tools to create any kind of application that interconnects objects.
<b>Q6</b>	Supported in its previous experience with the usage of other tools, this editor provides more simplicity for the generation of this kind of applications.
<b>Q7</b>	Internet of things and smart objects could benefit of this tool.
<b>Q8</b>	This editor provides an easy and intuitive form to interconnect devices.
<b>Q9</b>	This editor could be focused at the developing of other kind of similar applications.
<b>Q10</b>	The user makes fewer mistakes while works faster and efficiently by using this editor.
<b>Q11</b>	This editor can contribute to facilitate the development of services that provide interconnection of objects.
<b>Q12</b>	This editor can be considered useful and usable.

**Table 13 Survey given to the users**

To create the questionnaire we searched for twelve declarations. These asked for the user's opinion on the use of the tool, its possibilities and its possible impact in Internet of Things and Smart Objects.

To evaluate the results, we choose to show the responses of the users separately but at the same time we evaluated them together. We did that because it was interesting to evaluate from the point of view of both profiles. However, we intend to offer the same tool and the same functionality to both of them, that's why we evaluate them together.

## 5.2 *Results*

In Tabla 6 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).

Id	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
p01	5	4	5	5	5	5	4	5	5	5	4	4
p02	4	5	4	5	4	3	3	4	4	5	4	5
p03	4	4	3	4	3	4	3	4	4	4	3	3
p04	5	5	5	5	5	5	5	5	4	5	5	5

Anexo

p05	4	5	4	5	3	5	4	5	3	4	3	4
p06	5	5	4	5	5	5	5	5	5	5	4	5
p07	5	5	5	4	5	5	5	5	4	5	5	5
p08	4	4	3	4	3	4	5	5	4	5	4	4
p09	4	4	3	4	3	4	4	3	4	5	4	3
p10	3	4	4	3	3	5	4	4	3	5	5	4
p11	5	5	5	5	5	5	5	5	4	4	5	5
P13	5	4	4	5	4	5	5	4	5	5	4	4
P14	4	5	4	4	4	4	4	4	4	5	4	5
P15	5	5	5	5	4	4	4	4	5	5	5	5

Table 14 Responses of the users for each question

In Tabla 7 we present the descriptive statistics of all the set. Here, we can see the break down of each question: the minimum, the first quartile, the median, the third quartile, the maximum, the range, the rank between quartiles and mode. Figure 5 shows all this data in a Box and Whiskers Plot diagram.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
<b>Min</b>	3	4	3	3	3	3	3	3	3	4	3	3
<b>Quartile 1</b>	4	4	4	4	3	4	4	4	4	5	4	4
<b>Median</b>	4,5	5	4	5	4	5	4	4,5	4	5	4	4,5
<b>Quartile 3</b>	5	5	5	5	5	5	5	5	4,75	5	5	5
<b>Max</b>	5	5	5	5	5	5	5	5	5	5	5	5
<b>Range</b>	2	1	2	2	2	2	2	2	2	1	2	2
<b>Inter Qrt.-Range</b>	1	1	1	1	2	1	1	1	0,75	0	1	1
<b>Mode</b>	5	5	4	5	5	5	4	5	4	5	4	5

Table 15 Table with the general descriptive statistics

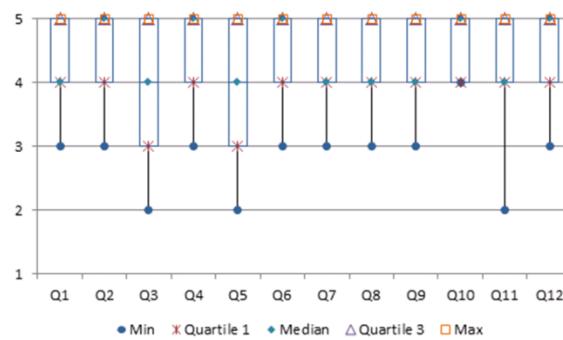


Figure 5 Box and whiskers plot for each question

By analyzing Tabla 7 and Figure 3, we can do the following interpretations:

Anexo

---

- Q10 has the highest minimum, in this case, 4 from a maximum of 5. This means that all the participants agree with the declaration, at the very least.
- Q2, Q4, Q6, Q10 and Q12 are the declarations with highest median. From this we can deduct that most of the participants agree with these declarations.
- Q10 is the only question with a rank of 1. This means that all the participants have the same opinion on this declaration. On the other hand, Q3, Q5 and Q11 have the highest rank. This indicates that there is a big difference between the answers of each participant.
- According to the mode, we can observe that Q5 has a mode of 3. This shows that the answer chosen by most of the participants is *Neutral*. On the other hand, Q2, Q4, Q6, Q7, Q10, Q11 and Q12 have a mode of 5, which indicates that the most chosen answer for these declarations was *Strongly Agree*.
- Regarding Q11, it has a rank of 3 and a minimum of 2, but the median is 4 and the maximum is 5. This shows that, despite having some low values, the declaration is considerably well-rated.

In Tabla 8 we can see the frequencies for the answers to each question. Here we have a break down of each question: the number of votes for each decision and the percentage corresponding to both of them. Figure 6 shows a bar graphic with the frequency of the answers in the set formed by all the profiles.

Anexo

Question		Strongly Desagree	Desagree	Neutral	Agree	Strongly Agree
Q1	#	0	0	1	6	7
	%	0%	0%	7%	43%	50%
Q2	#	0	0	0	6	8
	%	0%	0%	0%	43%	57%
Q3	#	0	0	3	6	5
	%	0%	0%	21%	43%	36%
Q4	#	0	0	1	5	8
	%	0%	0%	7%	36%	57%
Q5	#	0	0	5	3	6
	%	0%	0%	36%	21%	43%
Q6	#	0	0	1	5	8
	%	0%	0%	7%	36%	57%
Q7	#	0	0	2	6	6
	%	0%	0%	14%	43%	43%
Q8	#	0	0	1	6	7
	%	0%	0%	7%	43%	50%
Q9	#	0	0	2	8	4
	%	0%	0%	14%	57%	29%
Q10	#	0	0	0	3	11
	%	0%	0%	0%	21%	79%
Q11	#	0	0	2	7	5
	%	0%	0%	14%	50%	36%
Q12	#	0	0	2	5	7
	%	0%	0%	14%	36%	50%

Table 16 Frequency table for the general responses

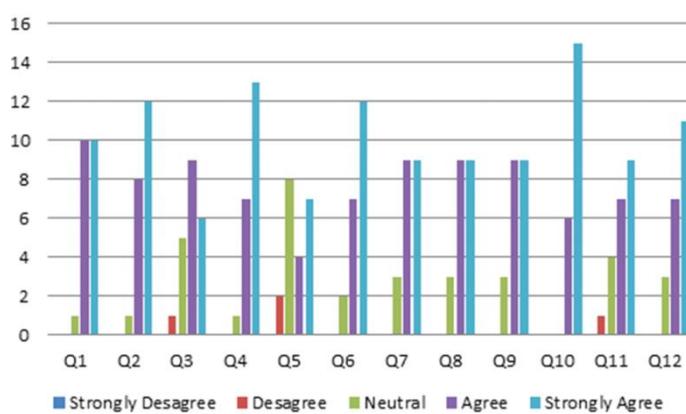


Figure 6 Overall response distribution

From Tabla 8 and Figure 6 we can figure things that couldn't be figure before. These are the interpretations:

- Q1 has a 48% of the votes for *Agree* and *Strongly Agree*, while only the 5% voted *Neutral*. In numbers this means 11, 11 and 1 vote, respectively, which indicates that the majority of the participants agree with this declaration.
- The Q7, Q8 and Q9 have a 43% of the votes for *Agree* y *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 is indecisive or doesn't believe in these declarations.

## 6 CONCLUSIONS

In this paper we describe a possible way to connect heterogeneous objects. We have also designed a Domain Specific Language, along with a graphic editor, to make this task easier. For that, we created the Midgar platform. Through the DSL we managed to abstract the problem of having to create an application in a programming language without needing programming knowledge. The user only needs to know the domain of the problem and use the DSL, which is generated by the editor. With this, we intended to make easier to create this kind of applications.

According to the surveys, the participants agreed that the editor is useful, efficient and allows the user to make less mistakes and work faster. They also highlight that it minimizes the complexity of the development for this type of applications. However, despite agreeing in many things, there's still people that is not sure if the editor could contribute to interconnect objects in an intuitive and easy way and if it could be used to generate other similar applications. Regarding the understandable creation of applications with this editor, as well as the tools it delivers, there is a great heterogeneity of responses.

There's also a great diversity of options about the possibility of the appearance of new object-interconnection services thanks to this editor. However, the opinions are mostly favorable.

The created DSL is independent from the platform that generates the applications and the editor itself. That's why it can be used as a generic base to create other editors and generators of applications that interconnect heterogeneous devices or create similar applications. Also, the applications consider that, with the help of a web graphic editor like the one from this research, we could favor the appearance of services that offer object interconnection. With this research we intended to prove that is possible to create applications to interconnect heterogeneous objects in a quick and easy way, even by user that are not software developers

## 7 FUTURE WORK

Internet of Things is part of the future and there's still so much 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:

- **Frameworks:** Create frameworks for each platform we want to give support to. This way we allow inexperienced users to create their own services and actions in his objects in a quicker and easier way, so they can connect them to Midgar.

- **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.
- **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 about different implementations of the IoT platform to check which one offers a bigger scalability.
- **Storage of data (SQL vs NoSQL):** Studies and tests about what kind of database is better to be used in IoT from the point of view of the efficiency and scalability.
- **Text editor with support for natural language:** Through this implementation any user should be able to create an application, just by describing what he wants.
- **Interaction and management of robots:** Add support to interact with robots and control them just like we do with a mobile phone and an object, but with the difference that these are able to follow an itinerary.
- **Support for more devices and sensors:** Offer support for a bigger number of devices that have sensors or can be connected to them.
  - Bigger number of Smartphones: iOS, Windows Phone, Ubuntu for Phones, Firefox OS.
  - Other types of micro-controllers: Raspberri Pi.
  - Other systems: GNU/Linux, Os X, PS3, PS4, Xbox, cars...
  - Integration of Smart Labels: RFID y NFC.
  - Sensors: humidity, proximity, presence, potentiometers, flex sensors, soft potentiometers...

## 8 REFERENCIAS

- [1] Akyildiz, I. W. S. Y. S. E. C., A survey on sensor networks, *IEEE Communications Magazine* (2002) 102–114.
- [2] Atzori, L., Iera, A., and Morabito, G., The Internet of Things: A survey, *Computer Networks* **54** (2010) 2787–2805.
- [3] Atzori, L., Iera, A., Morabito, G., and Nitti, M., The Social Internet of Things (SIoT) – When social networks meet the Internet of Things: Concept, architecture and network characterization, *Computer Networks* **56** (2012) 3594–3608.
- [4] Broring, A., Maue, P., Malewski, C., and Janowicz, K., Semantic mediation on the Sensor Web, in 2012), pp. 2910–2913.
- [5] Bulusu, N., Estrin, D., and Girod, L., Scalable coordination for wireless sensor networks: self-configuring localization systems, in *Proc. of the 6th International Symposium on Communication Theory and Applications (ISCTA '01)*, Ambleside, UK, 2001), pp. 1–6.
- [6] Deursen, A. Van, Domain-specific languages versus object-oriented frameworks: A financial engineering case study, *Smalltalk and Java in Industry and Academia ...* (1997).

Anexo

---

- [7] Deursen, A., Van Klint, P., and Visser, J., Domain-specific languages: an annotated bibliography, *ACM Sigplan Notices* (2000).
- [8] Dijkstra, E., The humble programmer, *Communications of the ACM* **15** (1972) 859–866.
- [9] Gama, K., Touseau, L., and Donsez, D., Combining heterogeneous service technologies for building an Internet of Things middleware, *Computer Communications* **35** (2012) 405–417.
- [10] García-Díaz, V., Fernández-Fernández, H., Palacios-González, E., G-Bustelo, B. C. P., Sanjuán-Martínez, O., and Lovelle, J. M. C., TALISMAN MDE: Mixing MDE principles, *Journal of Systems and Software* **83** (2010) 1179–1191.
- [11] Georgitzikis, V., Akribopoulos, O., and Chatzigiannakis, I., Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks, **10** (2012) 1686–1689.
- [12] González, E., Fernández, H., and Díaz, V., General purpose MDE tools, *IJIMAI* **1** (2008).
- [13] Groves, R. M., Fowler, F. J., Couper, M. P., Lepkowski, J. M., Singer, E., and Tourangeau, R., *Survey Methodology*, (Wiley-Interscience, 2004).
- [14] Gu, H., and Wang, D., A Content-aware Fridge Based on RFID in Smart Home for Home-Healthcare, in 2009), pp. 987–990.
- [15] Guinard, D., and Trifa, V., Towards the web of things: Web mashups for embedded devices, *Workshop on Mashups, Enterprise Mashups and ...* (2009).
- [16] Guinard, D., Trifa, V., Pham, T., and Liechti, O., Towards physical mashups in the Web of Things, in *2009 Sixth International Conference on Networked Sensing Systems (INSS)*, (IEEE, 2009), pp. 1–4.
- [17] Han, C., Jornet, J. M., Fadel, E., and Akyildiz, I. F., A cross-layer communication module for the Internet of Things, *Computer Networks* **57** (2013) 622–633.
- [18] Hao, C., Lei, X., and Yan, Z., The application and Implementation research of Smart City in China, in 2012), pp. 288–292.

Anexo

---

- [19] Hribernik, K. A., Ghairi, Z., Hans, C., and Thoben, K., Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, in 2011), pp. 1–9.
- [20] IoBridge, Thingspeak, (2013).
- [21] Kent, S., Model Driven Engineering, *COMPUTER-IEEE COMPUTER SOCIETY-* **2335** (2002) 286–298.
- [22] Kitchenham, B., and Pfleeger, S. L., Principles of survey research part 2: designing a survey, *IGSOFT Softw. Eng. Notes* **27** (2002) 18–20.
- [23] Kortuem, G., Kawsar, F., Fitton, D., and Sundramoorthy, V., Smart objects as building blocks for the Internet of things, *IEEE Internet Computing* **14** (2010) 44–51.
- [24] Lee, G. M., and Kim, J. Y., Ubiquitous networking application: Energy saving using smart objects in a home, *2012 International Conference on ICT Convergence (ICTC)* (2012) 299–300.
- [25] Likert, R., A technique for the measurement of attitudes, *Archives of Psychology* **22** (1932) 1–55.
- [26] LogMeIn, Xively, (2013).
- [27] McFarlane, D., Sarma, S., Chirn, J. L., Wong, C. ., and Ashton, K., Auto ID systems and intelligent manufacturing control, *Engineering Applications of Artificial Intelligence* **16** (2003) 365–376.
- [28] Meyer, G. G., Främling, K., and Holmström, J., Intelligent Products: A survey, *Computers in Industry* **60** (2009) 137–148.
- [29] Núñez-Valdez, E. R., Sanjuan-Martinez, O., Bustelo, C. P. G., Lovelle, J. M. C., and Infante-Hernandez, G., Gade4all: Developing Multi-platform Videogames based on Domain Specific Languages and Model Driven Engineering, *International Jorunal of Interactive Multimedia and Artificial Intelligence* **2** (2013) 33–42.
- [30] Pintus, A., Carboni, D., and Piras, A., The anatomy of a large scale social web for internet enabled objects, *Proceedings of the Second International Workshop on Web of Things - WoT '11* (2011) 1.
- [31] Pintus, A., Carboni, D., and Piras, A., Paraimpu, (2012).

Anexo

---

- [32] Pintus, A., Carboni, D., and Piras, A., Paraimpu: a platform for a social web of things, in *International World Wide Web Conference Com- Mittee (IW3C2)*, 2012), pp. 401–404.
- [33] Pintus, A., Carboni, D., Piras, A., Giordano, A., and Elettrica, I., Building the Web of Things with WS-BPEL and Visual Tags Web of Things using Service-oriented Architecture standards, in *The Fourth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2010)*, Florencia, Italy, 2010), pp. 357–360.
- [34] Piras, A., Carboni, D., Pintus, A., and Features, D. M. T., A Platform to Collect , Manage and Share Heterogeneous Sensor Data, in *Networked Sensing Systems (INSS)*, 2012), pp. 1–2.
- [35] Reed, A. B., Halpern, V., and Starr, J. E., Little languages: Little maintenance?, (1996).
- [36] Roman, R., Zhou, J., and Lopez, J., On the features and challenges of security and privacy in distributed internet of things, *Computer Networks* **57** (2013) 2266–2279.
- [37] Rothensee, M., A high-fidelity simulation of the smart fridge enabling product-based services, in *3rd IET International Conference on Intelligent Environments (IE 07)*, (Iee, 2007), pp. 529–532.
- [38] Selic, B., MDA manifestations, *The European Journal for the Informatics Professional*, ... (2008).
- [39] Tan, L., Future internet: The Internet of Things, *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)* (2010) V5–376–V5–380.
- [40] Telefónica, F., *Fundación Telefónica*, 2013).
- [41] The US National Intelligence Council, *Six Technologies with Potential Impacts on US Interests out to 2025*, 2008).
- [42] Wong, C. D. Mf. A. A. V. A., The intelligent product driven supply chain, in *Systems, Man and ...*, 2002).

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

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

**Cristian González García**

*University of Oviedo, Oviedo, Spain*

**Jordán Pascual Espada**

*University of Oviedo, Oviedo, Spain*

## ABSTRACT

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

## INTRODUCTION

Actually, most of the people have Smartphones or other intelligent mechanisms, like Smart TVs, Smart Labels (NFC, RFID)... This opens new doors to know the Internet of things.

Thank to these mechanisms, users can receive totally personalized notifications almost instantly, through an application, a message, a call, a code or by consulting a web service. Also, this gives the user the possibility to interact with other mechanisms in a much more easy way. For example, he can send a photo from his tablet to a Smart TV with a simple movement of the photo, ‘throwing’ it towards the Smart TV.

All this give both the user and the developer endless possibilities. In our case, with a view to process automatization, notifications, references and configurations. All this is achieved by the interconnection and identification of objects using their different sensors and protocols. This way, objects can interact among themselves, with or without the user’s interaction, but always performing tasks in an easier way or even automatically, according to the user’s preferences.

For example, if this is applied to food, information about it can be given to other mechanisms in a simple way. A fridge would be able to send mails to the cellular phone, informing about the amount of food stored inside or the products that are about to expire, thanks to the reading of the information (Rothensee, 2007). All this is possible if the food has a Smart Label (NFC or RFID) and the fridge has a reader, a computation unit that can perform the suitable actions and an Internet connection (Gu & Wang, 2009).

In the same way that it is applied to a fridge, this technology can be used with other elements, as the house itself and its conditions when we are outside, cities, supermarkets and shops to improve the way to bring their inventory or give information to their clients... all that is needed is to joint those smart object by using the suitable sensors.

Internet of Things doesn’t just present small-scale utilities at houses and shops but there are also some systems and IOT initiatives that include buildings and even integral cities.

Anexo

---

Madrid, Santander, Málaga, Barcelona, Luxembourg, Aarhus, Turku, Beijing... (Vienna, 2013) All of these are smart cities (Hao, Lei, & Yan, 2012) and they use sensors and others smart things in order to perform different tasks.

Humidity, temperature, ozone, movement, pressure capacity, gas, noise, light, pollution... There are different kinds of sensors that can obtain real-time information about the city's condition.

Traffic, parking, timetables about Smart city transport (Falvo, Lamedica, & Ruvio, 2012), environmental danger, lack of trash recycling, the quality of the water, light control, traffic control, reduction of CO<sub>2</sub>, energy saving, access to hospitals...All this is possible thanks to the combination of Smart Object and sensors in the cities.

For example, when a certain temperature is reached, the air-conditioning can be activated or the windows can be opened if it is not windy or rainy outside. Also, this information can be sent to Internet and the users within the affected zone can be notified by sending information or using smart labels that send the information to users with nearby compatible mechanisms.

By using other sensors (like those that react to movement) and the consignment of information to a process station and the cloud, both the traffic and the parkings can be controlled, and they also can offer or send information to the citizen, who will decide how they want to move.

In order to create a smart city, not everything has to be based on ordinary things, private things, can be also included. Also, each citizen can provide several sensors and install them in different places of their houses. For example, temperature and humidity sensors on a balcony. Other citizen could as well get access to this data if it is public and, following the same system, a page where the city map is selected, it is possible to observe the exact temperature and humidity in each zone. All this could extend and, with the use of the fridge mentioned before, know when certain element is needed (like water in a drought) and so the country could establish measures for any kind of situation.

This brings one problem: not all people have knowledge about sensors nor computers to be able to create the program that is needed for interconnection. Because of this, it's necessary to offer a system that can make this difficult task accessible for persons without specific knowledge about sensors and computing. It is necessary to offer it to persons with knowledge about language command, who know what they want to do and, thanks to a simple assembly and a little, easy configuration, can offer the information that they wish.

Anexo

---

However, applications created for this case or other similar to this one, as is normal, share some identical program points, they treat similar data and they behave in similar ways. But in spite of that, they are applications created by different people so in the end the inside is different too.

This will result in the existence of several applications that are practically the same and aim to resolve the same problem, but they are used in a different way. Also, they could have been developed in a different way and each one has required a great amount of time for that development.

However, the most important thing for users is to have a great speed and automatisms that make easier to perform a certain task. In this case, connecting the sensor and registering it.

This way, it arose the Model-driven engineering. This fact offered the necessary abstraction layer to create some languages of specific control and improve the usability and accessibility for the creation of applications for those people unconnected to the computer world or just making the use of certain technologies easier. They only need to know the domain.

The productivity and speed were increased in those areas in which the Model-driven engineering was applied.

In cases like this, the creation of a sensor net with its configuration and the connections with other mechanisms or smart objects is an arduous task. The diversity of objects (telephones, computers, smart labels, sensors...) and the different kinds of connection (Wi-Fi, Wireless, Bluetooth, SMS,...) make administration very difficult. All this, along with the different kinds of sensors (humidity, temperature, pressure, speed, movement...) makes this application, very hard to use for most of the people.

Because of this, the objective is the application of the Model-driven engineering to the field of sensors, and, by doing so, being able to form and create sensor nets easily and allowing the user to choose what objects he wishes to interact with and the actions that he wanted to perform in a clear and easy way.

In this way, we get to reduce the complexity thanks to the layer abstraction that is inserted to represent a specific domain. As well, the portability, interoperability and reusability created by it are improved.

The objective of this research is the presentation of the basics and the idea that will serve as a base to create the next prototype. This one will consist of a platform web in which smart object and sensors can be registered through the use of Domain Specific Language (DSL) done with Graphical Modeling Framework (Kolovos, Rose, Paige, & Polack, 2009) and Textual Modeling Framework.

With the creation of this platform, Muspel, we want to solve several existing problems. First, there is the difficulty to interconnect and communicate different platforms, smart objects and existing sensors. Nowadays, because of the many different software and hardware, it's necessary to have some technical knowledge about each element, and a specific application for each mechanism. The problem we want to solve is how to make this easy for any kind of user, because all this requires a very technical knowledge and so it's limited to just a small percentage of people. That's why, by using a framework, we try to make the interconnection of heterogeneous smart objects totally accesible for any user who knows the smart objects.

An example is the creation of a sensor net and its configuration in a graphic form. It is as simple as dragging boxes and writing the graphics we need and we ask for. By doing this, any person, regardless of their computer skills, could create an application based in a minimum knowledge of control.

This makes the same application a good one for modeling, too, without having to worry about different kinds of mechanism connections, because the user would not have to take into account if the mechanism is Bluetooth, Wireless, infrared, Wi-Fi or HTTP.

All things created with this model are abstracted out of all short and medium configuration processes. In this way, the user -who only knows about the model control- can create sensors and different applications which administer different actions according to the user's condition.

## CONTRIBUTION

With this research we intend to make the following contributions:

**Simplicity of software:** it will be offered in a easy and intuitive form, exact or graphic, like the user wants, to create the applications. In this way, the user will only have to know each element (sensor, action, smart object...) and the different configurations he can apply to them, that is, if it is allowed. This way, any user, even without having knowledge in programming or electronics, could create and register an application based in sensors.

**Reutilization of used technologies:** A lot of programs that use the same software can be done.

Creation of an abstraction layer in a bigger level: to configurate and work with different connection kinds in a low level is something to avoid, for example Bluetooth, Wi-Fi, Wireless, SMS, HTTP,... All

this will be done clearly and the user will only have to choose the connection that he wants to perform with the object, if this one can be accessed through several connections.

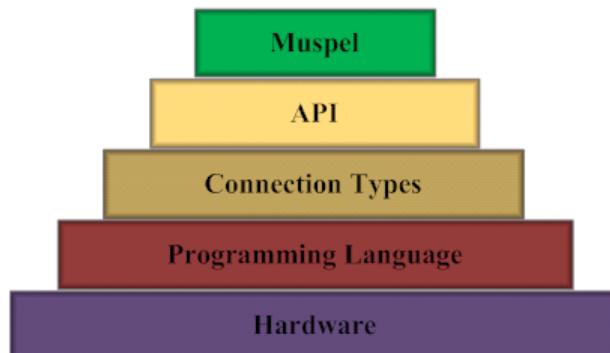
**Domain Specific Language for creating of applications:** A domain specific language will be created to serve as a base for creating applications based in sensors.

**Domain Specific Language for exchange of data:** A domain specific language will be created to serve as a base for exchanging data between smart objects.

**Metamodel:** A specific model that serve will be created as a base for creating some models for applications based in sensors and which want to solve similar problems.

**Data centralization:** A system that will be able to centralize the developed applications will be created. This way, any user will be able to get access to public content published by other people.

**Connection abstraction:** Thanks to the application, we will abstract and encapsulate all connections and communications of objects to Internet. This is because all objects that wants to send and receive data might connect or perform a necessary request to the application. This way it won't be necessary to use other protocols that can be slower, spend less battery or aren't available.



*Figure 1. Muspel's Application Layers*

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

## STATE OF THE ART

Smart cities are the pinnacle of sensor nets. In them the situation of certain zones is watched and also certain mechanisms act accordingly.

The most classical example are the lights. These ones turn on when the sensor doesn't receive a minimum amount of light previously fixed.

By implementing this system we could make possible for the smart city to send a text message to the subscriber's cellular and inform about a certain situation. For example, the traffic on the road. This example can be extended to parkings where we are notified about free places even before entering.

These configurations are created by the government or specific people thanks to a certain software. All this is not accessible for users who cannot add a sensor to the government's net.

However, there are certain web sites that offer users the possibility of sharing the measurings done by their own sponsors. This is the case of COSM (previously known Pachube) (LogMeIn, 2013). Other examples are Paraimpu (Piras, Carboni, Pintus, & Features, 2012) o ThingSpeak (IoBridge, 2013), etc.

COSM offers its API and several advices to work over the mother base Arduino. Despite this, the user will have to learn how to use the API and work in a lower lever with Arduino, because it is programmed in a programming language C. It offers data in different forms: XML, JSON, CSV. We can access to them by a free and bidirectional API RESTful (upload and download of data) available in several programming languages (C, Ruby, PHP, JavaScript, ...), as in modifiable graphics in the web. It allows users to upload their applications in a public or private way and a total administration of them and their data. It also allows users to chat in the same page where the sensors are.

ThingSpeak offers the same than COSM, but there is a difference. It offers an only API for connecting to the service and it doesn't give API to work with any system, like Arduino.

There are pages and a lot of different mechanisms to interact according to conditions. Among them, there are some which are intelligent. The most used nowadays is Smartphone. But Smart TVs, tablets, NFC, RFID and also the laptops are getting more presence at homes, in the streets or at establishments. It can interact with each one in several ways: Wi-Fi, Wireless, infrared, Bluetooth, SMS, call, MMS, HTTP, SMTP, IMAP,... The only problem is connecting with them. Sometimes it is simple, for example when

---

Anexo

---

we send an electronic message. Other times it can be more complex, as sending a SMS, MMS or connecting Bluetooth; this is due to the absence of information, examples or the system complexity.

Because of these problems, It is needed to know the sphere and the different programming languages, technologies and mechanisms. A lot of knowledge is required just to do a few things.

Because of that, with the introduction of the Model-driven engineering we intend to make all this process easier for the user thanks to the introduction of an abstraction layer. Users will have to know the sphere and have a little knowledge about each tool. However, this last process will be drastically simplified to offer a very easy system for creating sensor nets next to conditions that are necessary in a certain act.

Our goal is to create a web site that centralizes all the information sent by users as well as the register of its applications. We will provide a RESTful API, too, so a download data can be sent from different notations. We will also provide a framework to make possible to create applications in an easy way for different systems, like Smartphones with sensors and sensor base plates. By doing this, the user, just by using this framework, would be able to create a native application that uses the Smartphone's sensors or base plate and send the desired information to the web. To send all data a standard DSL will be created, one that will be compatible with any application or any sensor. With all this we intend to make the extension through new systems and sensors quicker and easier.

## **DESCRIPTION**

### **Muspel**

Muspel will be established from the web site and from different APIs and frameworks. The web site will centralize all user information and his application's register. It will be responsible for managing this information and, via RESTful service, providing the necessary data required by the applications.

To make the interaction easier for the user, we will put a set of APIs with which he could connect to the service and some frameworks which could be used to develop the application in an easier way, from Smartphone to base plates.

Anexo

---

All the available information would be accessible in different forms, like XML, JSON y CSV. In spite of that, for direct communication with applications, it would be necessary to create a standard DSL that could serve to send data. This way, the future extension to more systems and sensors would be easier.

## **System & equipment**

The main piece for connecting sensors to the environment will be a mother base Arduino (Hribernik, Ghrairi, Hans, & Thoben, 2011). The required sensors of data and several mechanisms are used to perform actions like loudspeakers, LED's, motors, servomotors... These will be connected to the mother base Arduino (Yamanoue, Oda, & Shimozono, 2012).

The Arduino will be connected to the PC via USB on the first tests: then, through a Wireless unit (Georgitzikis, Akribopoulos, & Chatzigiannakis, 2012). For the performing of all the tests there will be an exclusive computer that will process and save all the data, and it will make the connection with different systems and smart objects.

Several Smartphones will be used, each one with different operating systems like smart objects, sensors, and text messages, multimedia messages, calls or mails according to the configuration.

Other kind of smart objects which will be used are Smart TVs, sensor nets, tablets and computers.

## **Software**

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

We will use the Java program language; that will create an application which will be easily ported to different known systems. The only requirement is to install the Java virtual machine so it can work with the application.

To create the model it an Ecore metamodel will be used.

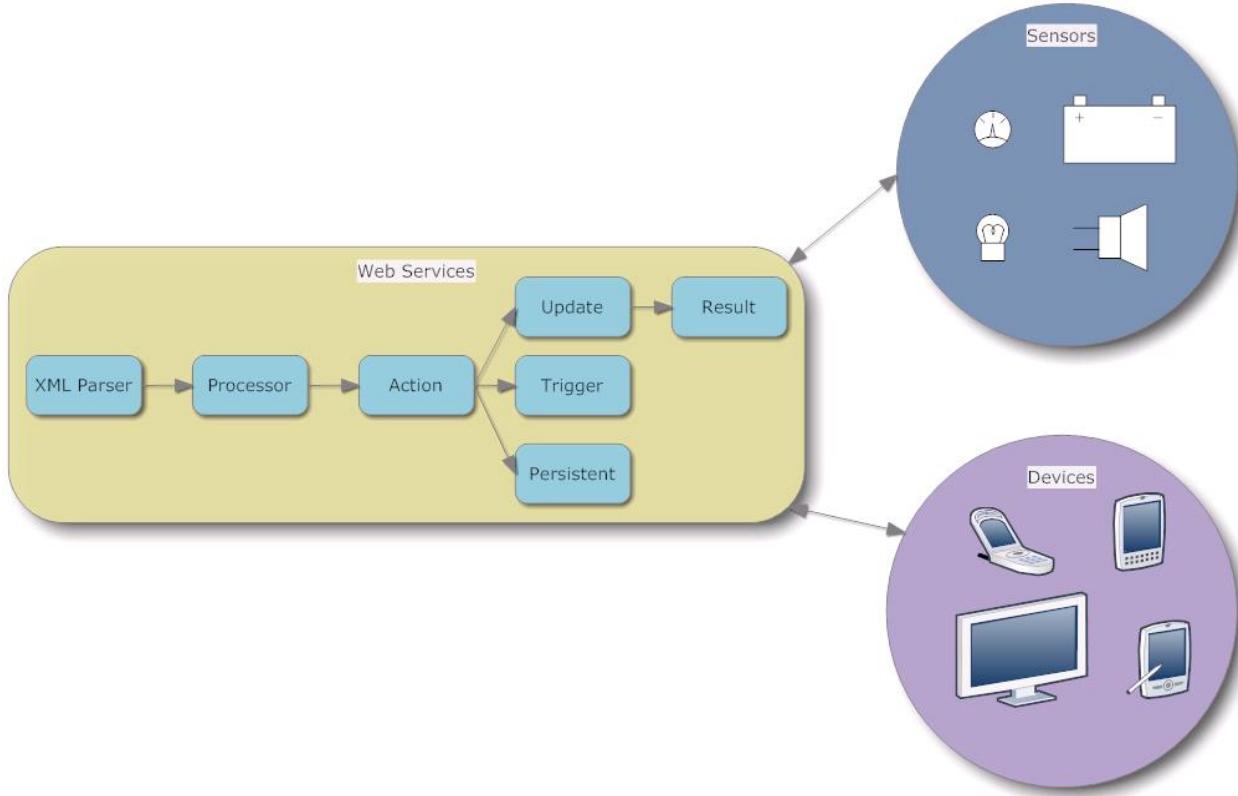
Anexo

---

Herewith it wants to create a framework to implement applications in a simple way in cellular phones or well to use frameworks of the graphic or textual interface. If the first one is used, a user with programming knowledge should be able to create a new design and implement it. This way, he could use it in a lower level and make a deeper configuration of the application. With the last two, any user without programming knowledge could implement an application because the domain is the only thing he needs to know. With the framework of graphic interface, the user only might drag and drop boxes and fill his data to generate the application. By using textual framework, the user would have to write, according to some rules which would be given in real time, the application in each text.

## **Architecture**

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



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

First of all, to explain that the way to interact with the web service that is sending the text in XML format,

we will describe the internal operations of the web service. This has a series of steps:

When a XML file is received, it processes that one en the XML Parser platform. Once it processes the file, it sends the data to the processor. This one, from this data, knows if it is a sending request of data or an order, and then it runs a series of actions: Update, Trigger and Persistent. All three will be run at the same time.

If the persistence configuration is activated, one of these actions will be kept (or not) within the data base.

Anexo

---

If the processed action must break out a process, the Trigger will be the one which makes it. For example, when a temperature sensor shows 100°C, it will send a call to the firefighters and an e-mail to the cellular of the sensor's owner.

At last, the Update executes the necessary update in the web service and in the respective URL; it will always keep the RESTful service. It creates, erases or updates new files when necessary; also, new URL, images or folders...

Once the last process I finished, the final result will be returned. This can be a XML with successful confirmation of the processing of the process trigger. If data or one or more text files are sent with the order, according to the asked format, as well as the respective XML to send the required orders.

## **Applications**

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

Anexo

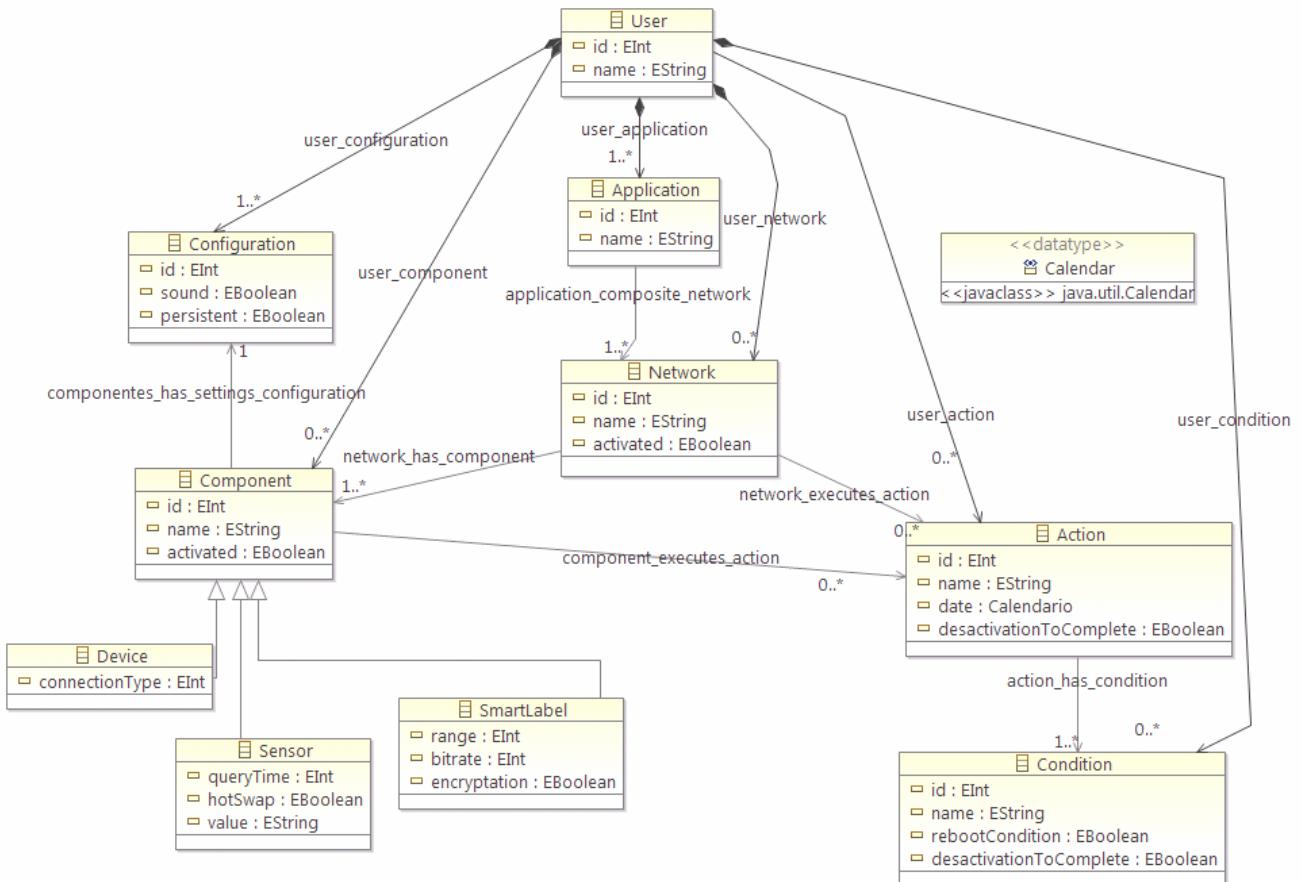


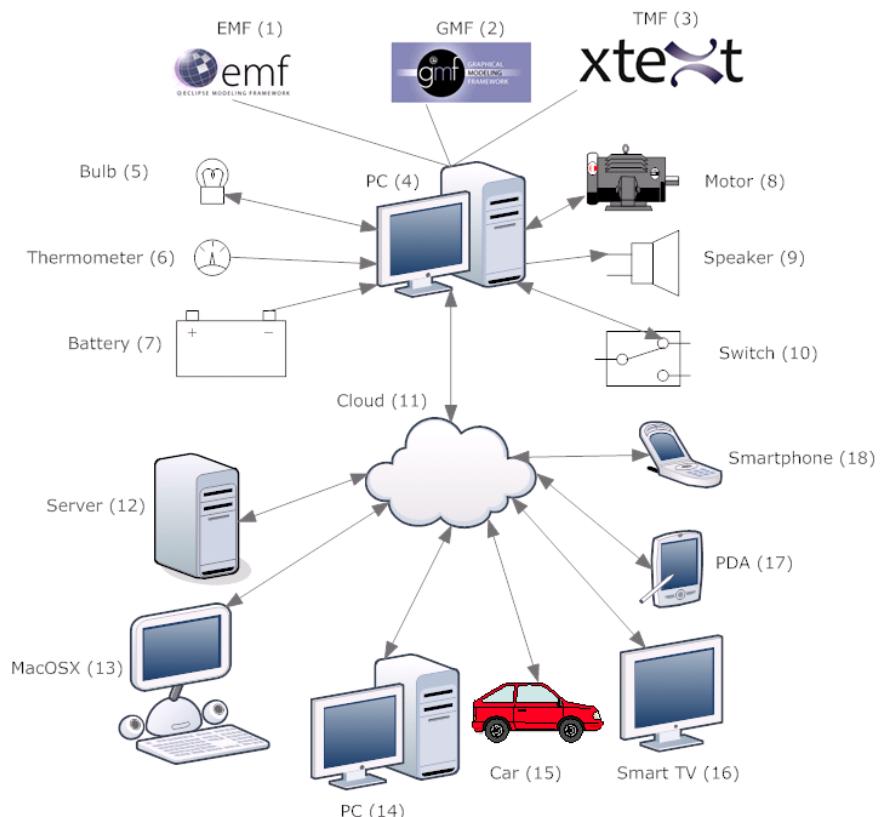
Figure 3. Ecore metamodel used to describe elements, connections and limitations of modeling language that will be used.

After that, we need to create all the elements required to abstract whatever we want through the model. That is, different kinds of connection, options, configurations, etc.

In other way, its temperature, humidity, movement and light sensors will be connected to the mother base Arduino; as well as mechanisms like loudspeakers, LED's, a motor, a servomotor and a relay.

Next step is programming sensors so they can receive the wanted values at the time we want.

After that, the Java model has to be modified to be connected to Arduino and, this way, the configuration that was created as a model interacts directly with the hardware, so the data can be processed in both directions: from model to Arduino and vice versa.



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

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

Anexo

---

- Bulb (5): It has bidirectional connection. We can know if it is switched on or switched off and also turn it on or turn it off. Because of that we can, under certain conditions, turn it on or turn it off or know if we need it to be in the state in which it is at that moment.
- Thermometer (6): We can only check its state, that is, the temperature that it gives us. This one can be located at home, outside or in a mobile mechanism.
- Battery (7): We can only check its state, that is, the remaining battery. For example, in a cellular phone or a laptop.
- Motor (8): It has a bidirectional connection. We can know if it is switched on or switched off and turn it on or turn it off under certain conditions or by direct orders from an IOT mechanism. It can be the load motor of a compressor and, if it is detected that is full, it can stop or inform us and, this way, alter the power that is given to the battery load.
- Speaker (9): We only can give orders. In this case, it can be used to alert us of certain states. For example, when the battery is over 5% or when the motor does not work, it sends an alarm warning.
- Switch (10): Through its bidirectional communication we can know the state of this element and control it. For example, if this belongs to the heating, we can turn it on or turn it off depending on the exterior temperature and, even from the Smartphone (18) or the PDA (17), send an order to turn it off until the home thermometer has reached 21°C.
- Cloud (11): The cloud is Internet. It is a resource which connects all mechanisms, all the data is sent through it, and it is also the calls to other mechanism are performed.
- Server (12): Server that can receive and send data. This one can be used to keep mechanism data and send e-mails to different mechanisms when they meet a certain condition, like the compressor and the battery being full and the temperature surpassing 45°C.
- Mac OS X (13): A computer with an OS X operative system that can interact in a bidirectional way with the cloud, that is, with different mechanism connected to it.
- PC (14): A computer with a Microsoft Windows or a GNU/Linux operative system that can interact in the same way as the Mac OS X.
- Car (15): Car with bidirectional connection, that car transmits its GPS position in a public or private form, even other data and emergency signals. Also, it can receive notifications of other IOT mechanisms. For example, from another car that has had an accident and it is within 1 KM.
- Smart TV (16): This device has a bidirectional connection. Thanks to it, video calls or VOIP calls can be sent and received.
- PDA (17): By PDA the data can be sent and received to operate with different mechanisms. For example, if we notice that our house temperature sensor shows 10°C, we can send an order to turn on the heat.
- Smartphone (18): As the same way than PDA (17), we can interact with other smart devices, as well as receiving the GPS position of the car (15) in the case of an accident.

## EXPERIMENTS

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

Anexo

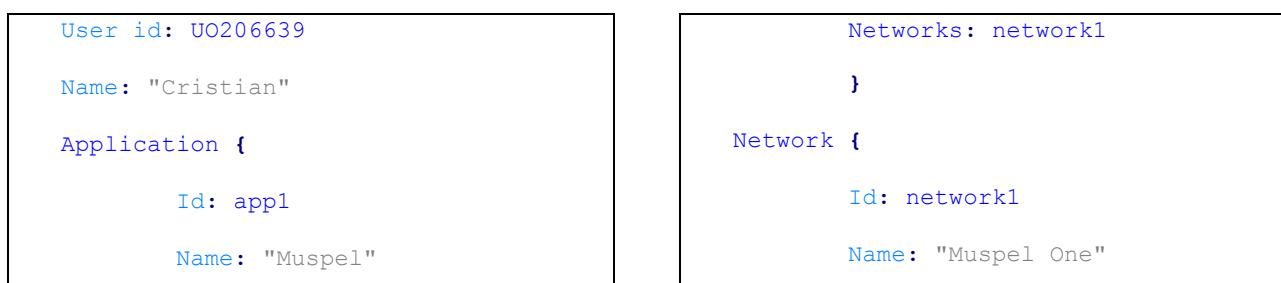
---

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

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

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

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



Anexo

```
Activated: TRUE
Components: device1
sensorTemp sensorHum sl4

Actions:
actionCallFirefighters

}

Device {

    Id: device1
    Name: "Smartphone 1"
    Activated: TRUE
    Connection Type: "SMS"
    Configuration: conf1
    Actions:
    actionCallFirefighters
}
Sensor {
    Id: sensorTemp
    Name: "Temperature"
    Activated: TRUE
    Query Time: 10000
    Hot Swap: TRUE
    Configuration: conf1
    Actions:
    actionCallFirefighters
}
Sensor {
    Id: sensorHum
    Name: "Humidity"
}

Activated: TRUE
Query Time: 20000
Hot Swap: FALSE
Configuration: conf1

Actions:
actionCallFirefighters

}

SmartLabel {

    Id: sl4
    Name: "NFC Active"
    Activated: TRUE
    Range: 50
    Bitrate: 30
    Encryption: FALSE
    Configuration: conf1
    Actions:
    actionCallFirefighters
}
Configuration {
    Id: conf1
    sound: TRUE
    persistent: TRUE
}
Action {
    Id: actionCallFirefighters
    Name: "Call firefighters"
    Date (dd/mm/yy): "13/12/12"
```

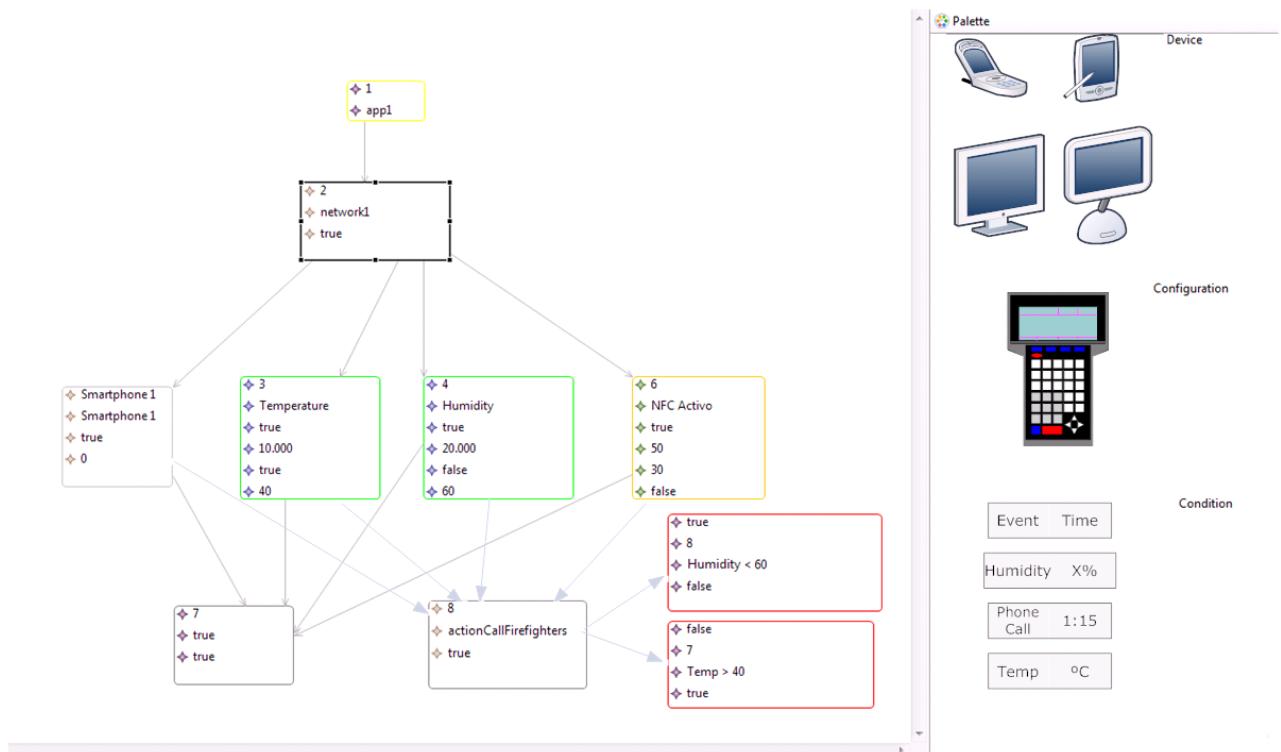
Anexo

---

```
Desactivation to complete:  
TRUE  
  
    Conditions: conditionTemp40  
conditionHum60  
  
    }  
  
Condition {  
  
    Id: conditionTemp40  
  
    Name: "Temperature > 40"  
  
    Reboot Condition: FALSE  
  
    Desactivation to complete:  
TRUE  
  
    }  
  
Condition {  
  
    Id: conditionHum60  
  
    Name: "Humidity < 60"  
  
    Reboot Condition: TRUE  
  
    Desactivation to complete:  
FALSE  
  
    }
```

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

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



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

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

## FUTURE WORK

Future work that could be done would be:

- The mobile mechanisms that are connected to the platform need a framework which makes the synchronization and communication with the platform. To make possible for several electronic mechanisms with different features and different operative systems to connect to the platform it's required that this connecting application is implemented in several programming languages; this way the application that adapts better to the specifications could be installed in each mechanism. Because of that, part of the future work could be to create native frameworks in other platforms and in other languages, like: C++, C#, Ruby, PHP, JavaScript, Objective-C, ...
- To complete the specific language capacities of graphic domain to allow the inexperienced users to specify business processes which involve the combined work of several smart objects and mechanisms. To develop this new specific language of graphic domain we want to evaluate some languages and tools designed to model business processes.
- Thanks to the heterogeneous nature of mechanisms with its own operative systems and its different functioning, it would be interesting to expand to a bigger number of systems: Smartphones (Ubuntu, Windows phone, Firefox OS), video consoles (PS3, Wii, Wii U, Xbox 360, Nintendo DS, Nintendo 3DS), computers (Microsoft Windows, Mac OS X, GNU/Linux), graphic card with sensors (NVidia, ATI), Smarts Cities...
- Actually, every mechanism has several forms of interconnection with the exterior. For example, a Smartphone or cellular phone with infrared, Bluetooth, *Short Message Service* (SMS), *Multimedia Messaging System* (MMS) and Wireless. A computer can have some of these and even connection ports, like Universal Serial Bus (USB), LPT1 ports o Ethernet ports. Other devices have High Speed Downlink Packet Access (HSDPA), (High-Speed Uplink Packet Access (HSUPA), Global System for Mobile communications (GSM), Worldwide Interoperability for Microwave Access (WiMax), Long Term Evolution (LTE), Universal Mobile Telecommunications System (UMTS), Code Division Multiple Access (CDMA),... Because of this many possible forms of data sending, it can be increased to give support to a lot of protocols and give more possibilities in the interaction with other mechanisms or serve as an extra support if there is a failure in one of the used objects.

## CONCLUSIONS

As it has been demonstrated, the way to abstract the configuration, organization and creation based in sensors has been taken to a very high level thanks to the use of model-driven engineering.

Users can choose between a textual environment that drive it or a graphic environment where they drop the boxes, fill them with data and joint them.

Among data, user only has to choose those one he prefers, choose the kind of connection, conditions, actions, persistence, and he won't have to take care of how it operates in the inside.

All that makes possible for unconnected people (as long as they know the sphere) to create applications of sensor nets for personal use in an easy and quicker way, which can be used and modified again. They can create a net at home or at work, or one net in each room and they can modify it in a little time. In the primary sector, they can create sensor nets to control fields, greenhouses, and automate tasks in a simply way, according to a mechanism that receives the action, for example, a motor or a robot.

## REFERENCES

- Falvo, M. C., Lamedica, R., & Ruvio, A. (2012). An environmental sustainable transport system: A trolley-buses Line for Cosenza city. *International Symposium on Power Electronics Power Electronics, Electrical Drives, Automation and Motion*, 1479–1485. Ieee.
- Georgitzikis, V., Akribopoulos, O., & Chatzigiannakis, I. (2012). Controlling Physical Objects via the Internet using the Arduino Platform over 802.15.4 Networks, *10*(3), 1686–1689.
- Gu, H., & Wang, D. (2009). A Content-aware Fridge Based on RFID in Smart Home for Home-Healthcare, *02*, 987–990.
- Hao, C., Lei, X., & Yan, Z. (2012). The application and Implementation research of Smart City in China, (70172014), 288–292.
- Hegedus, A., Horvath, A., Rath, I., Ujhelyi, Z., & Varro, D. (2011). Implementing efficient Model Validation in EMF Tools.
- Hribernik, K. A., Ghrairi, Z., Hans, C., & Thoben, K. (2011). Co-creating the Internet of Things - First Experiences in the Participatory Design of Intelligent Products with Arduino, (Ice), 1–9.
- IoBridge. (2013). Thingspeak. Retrieved from <http://www.thingspeak.com>

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

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

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

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

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

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

## KEY TERMS AND DEFINITIONS

Internet of Things: network of daily objects connected to each other to allow an interaction among them.

Model-driven engineering: methodology of software development that is focused on model creation near the concept of private domain instead of software.

Sensor: mechanism that is capable of detecting physic or chemical magnitudes. For example: temperature, light, humidity, CO<sub>2</sub>, radiation...

Sensor platforms: Set of sensors in a same platform to allow the process of interconnection and make it easier.

Sensor-based applications: applications created to work and interact with sensors.

Anexo

---

Smart objects: objects that can interact with others. For example, two Smartphones or a Smartphone with the TV.

Domain Specific Language: language given to solve or involve a specific problem and provide a technique for solving private situations.

Web Services: technology that uses a series of protocols and standards to exchange data between applications.

Smart City: City that creates an environment where different mechanisms are interconnects among them, all by using sensors and automations.

8.2 *Domain Specific Language for the development of Educative Multiplatform Videogames Case study GADE4ALL*

## **Domain Specific Language for the development of Educative Multiplatform Videogames: Case study GADE4ALL**

Cristian González García<sup>a\*</sup>, Edward Rolando Núñez -Valdez<sup>a</sup>, Juan Manuel Cueva Lovelle<sup>a</sup>, Oscar Sanjuán Martínez<sup>b</sup>, Cristina Pelayo García-Bustelo<sup>a</sup>

<sup>a</sup> University of Oviedo, Department of Computer Science, Sciences Building, C/Calvo Sotelo s/n 33007, Oviedo, Asturias, Spain. Tel: +34985103397

<sup>b</sup> University Carlos III of Madrid, Computer Science and Engineering Department, Avenida de la Universidad, 30, 28911, Leganés, Madrid, Spain.

<sup>a</sup> [gonzalezcristian@uniovi.es](mailto:gonzalezcristian@uniovi.es), [nunezedward@uniovi.es](mailto:nunezedward@uniovi.es), [cueva@uniovi.es](mailto:cueva@uniovi.es), [crispelayo@uniovi.es](mailto:crispelayo@uniovi.es)

<sup>b</sup> [oscar.sanjuan@uc3m.es](mailto:oscar.sanjuan@uc3m.es)

\* Corresponding author.

**Keywords:** Applications in subject areas, Human-computer interface, Interactive learning environments, Media in education, Programming and programming languages.

### **ABSTRACT**

Nowadays, educators are starting to show interest in educative videogames as a mean to motivate their pupils and improve their learning. However, the development of videogames requires quite a bit of knowledge in the programming area. Because of the special approach involved to make them more accessible and adaptable to the different levels of knowledge and gaming skills, educative applications are usually harder to create. In addition to this, the existence of various platforms makes this project too big for people that may not possess the required knowledge in this field, like educators. The goal of this research is to create a Domain Specific Language for the development of educative multiplatform videogames to help and motivate both teachers and students to take advantage of the new technologies to improve the way of teaching and learning. The objective is to make possible for people without

programming knowledge to generate an interactive, multiplatform educational environment in a fast and efficient way.

## 1 INTRODUCTION

The use of new technologies, like Smartphones, Internet, tablets or computers is widely spread. These are used both for personal use and learning and offer a wide range of possibilities (Lavín-Mera, Moreno-Ger, & Fernández-Manjón, 2008). Currently, mobile learning and the using of videogames to motivate the students to learn are experiencing a major increase (J. P. Gee, 2003). Many schools, high schools and colleges have digital screens, computer rooms and, in some cases, even computers or tablets for each student. Educators are also becoming more interested in these educative videogames to motivate the students and improve their learning (Ozcelik, Cagiltay, & Ozcelik, 2013; Papastergiou, 2009).

However, the teaching staff doesn't have specific applications for each subject or something to create them. The existing tools are very costly and quite complex to use, thus difficulting the creation of application (Lavin-Mera, Torrente, Moreno-Ger, Vallejo Pinto, & Fernández-Manjón, 2007). In this case, the generated videogames will serve an educational function rather than a commercial one. This way, the student will be motivated to study through gaming (Griffiths, 2002; Moshirnia, 2007; Tüzün, Yılmaz-Soylu, Karakuş, İnal, & Kızılıkaya, 2009).

In this research, we focus on the benefits that both teachers and students obtain with this tool. However, there are multiple scenarios where we could apply everything that is related to this project. For instance, the games could be aimed to specific users, like educative games for toddlers, between 0 and 4 years old, so they can learn basic things, like colors or the alphabet. Games could also be used to help old people or people with some kind of disabilities or illness, as (Griffiths, 2002) says about some researches on kids with Attention Deficit Hyperactivity Disorder (ADHD), Human Immunodeficiency Virus Infection (HIV) or diabetes. Other games can be aimed to stimulate specific capacities such as: memory, logic, eye-hand coordination, reaction time, self-esteem, spatial vision, etc. All this can be achieved through the use of videogames (Griffiths, 2002).

Besides, the games make possible to generate all the necessary contents for the study as well as the tests to confirm the acquisition of knowledge, so they positively contribute to preserve the environment. The wood cutting is reduced because there is no need for paper to print exams or notes, which contributes to reduce deforestation (She & Wang, 2011).

In order to perform this research, the use of Model Driven Engineering was proposed (Kent, 2002).

With this we aimed to increase the efficiency and solve software developing's typical problems (González, Fernández, & Díaz, 2008; Selic, 2008). In this proposal we suggest to use a Domain Specific Language (DSL) to develop Educatice Multiplatform Videogames in two dimensions (2D).

With the creation of the DSL we sought to create an abstraction based on certain notations and solve the problem of porting between platforms by using a single language and making it easier to use. This way, a teacher will only need a single specific tool and will also be able to create educative multiplatform games quickly and easily, with which the student would get access to tutorials, lessons and exercises to practice everything he has learned so far. The teacher will also be able to create exams and distribute them quicker among his students, who will receive their marks instantly. When creating these exams, the teacher will be able to combine images, text, video and sound to generate each question.

The rest of the proposal follows this structure: In section 2 we make an introduction to the problems that exist nowadays. In section 3 we explain the contribution that is being made with this research. In section 4 we analyze the current situation of the development of applications for mobile devices and make a brief explanation of Model Driven Engineering and the Domain Specific Languages, as well as an research on some existing videogame editors and the situation of videogames in the education.

Section 5 details the kind of questions that can be created in the games, the software used for this research, the proposed architecture and its implementation. Section 6 explains all the methodology and the case study used to perform the experiment. Section 7 shows the results of the evaluation. Section 8 contains the discussion of the research. Section 9 covers some of the possible areas for future researches. Section 10 lists the conclusions obtained through this research.

## 2 PROBLEMS

According to (J. P. Gee, 2003), one of the most effective ways of learning is through videogames, something recognized by cognitive science in recent researches. This way we can boost the students' motivation for learning through entertainment software, as proposed in (Moshirnia, 2007; Ozcelik et al., 2013; Tüzün et al., 2009). The biggest problem nowadays is

the difficulty of developing ubiquitous educative applications. This is due to the youth and fast evolution of mobile devices (Lavín-Mera et al., 2008). This offers the necessary calculation capacity to create educative videogames, but the high diversity of devices implies an extra difficulty when trying to develop for all of them in a simple and effective way (Lavin-Mera et al., 2007). This is because there is not a common specific language for all of them that allows to port the content to each platform. One possible solution would be to create a domain specific language to obtain that abstraction layer and encapsulate the domain of the problem in it (Arie Van Deursen, Klint, & Visser, 2000).

## 2.1 *Education*

Smartphones and laptops have an unlimited capacity and a better portability than physical books. That's why, having the portable lessons and the tests after these, the student will be able to take the tests as many times as he wants to learn, even in places with limited access to electronic devices (Kam & Rudraraju, 2007). Through the use of videogames, cooperation, competence and self-improvement will be stimulated (Griffiths, 2002). Thanks to its portability, the user will be able to move and interact with other people while doing the activity, thus making easier to solve a problem through cooperation (Lavín-Mera et al., 2008). This will have repercussions in the individual learning, by trying to surpass other students, and also in the team cooperation, by trying to find a combined solution to the problem. The student will also have the possibility of repeating the lessons and exams suggested by the teacher again and again. This will prevent the student from getting frustrated when failing, because he will be able to repeat the same lessons until he gets a good mark, which will positively improve his motivation for learning (J. Gee, 2005; Tüzün et al., 2009). Besides, exams can be performed with the generated educative games. The exams would be quickly distributed among the students, its correction would take just a moment and the students would receive their marks instantly, which benefits both them and the teacher.

## 2.2 *Videogame development*

Creating videogames requires people with experience in software development and a great knowledge not only on the domain of educative videogames, but also in the different programming languages that will be necessary to take the videogames to multiple platforms. To partially solve the first problem, derived from the software development of the application, Model Driven Engineering was applied. For the second problem, the main inconvenient lies in making possible for people without knowledge in informatics to develop educative videogames in a quick and simple way. (Lavin-Mera et al., 2007). The solution for this case resides in the

necessity of offering an application that allows them to perform that task. This way, teachers could create an educative videogame, which can contain notes and questions about the content of these notes, so the students can consult them anywhere, as long as they have downloaded the game previously.

### 2.3 *Ecology*

One of the big problems that concerns us humans the most is the pollution and deforestation, which results into an even bigger pollution, because it damages the lungs of the planet: the trees (She & Wang, 2011). Because of this, a great emphasis has been done in green technologies and low-power-consuming hardware, as well as in the saving of paper, the proper handling of wastes, the moderate deforestation and the humanity's awareness regarding all of this problems (She & Wang, 2011; Yang & Zhao, 2011). Our research helps this cause, as there is no need to print the notes, exercises and exams for the students.

This helps both the study centers and the students to save lots of paper and ink. This way, the consumption of paper and ink is reduced, which results into a reduction of the deforestation and a lesser consumption of chemical products in exchange for electric expenses. This can come from renewable energies, which results in lesser pollution and more respect for the environment.

## 3 CONTRIBUTION

With this research we intend to make significant contributions in fields like videogame development, education and ecology. Our goal is to offer speed and simplicity for the game development process while saving resources and reducing the pollution. We intend to solve all the problems mentioned previously through the following contributions:

- **The creation of educative videogames in a quick and simple way:** We offer the possibility of creating educative videogames in a quick and easy way through the use of an editor or the edition of the files that contain the DSL. All the menus and levels contained in the game will be fully editable.
- **Creation of multiplatform educative videogames in one single step:** Once the videogame has been created, it will be possible to export it to various platforms at the same time. It won't require any extra changes or modifications. The available platforms will be HTML5, iPhone, Android and Windows Phone.
- **Domain Specific Language for multiplatform support:** We offer a whole new Domain Specific Language (DSL) that is used to make multiplatform applications. This DSL is divided in two eXtensible Markup Language (XML) files, one for the interface and other for the questions. This way, we only have to change the data in the specific files, which will be reflected in every application regardless of the platform. This means a great flexibility and simplicity for the portability of the educative games.
- **Faster correction:** The results of this research can be used to create tests for the students, making possible to obtain the results immediately. Once the students finish

their exams, they instantly receive their marks, something convenient for both the students and their teacher.

- **Paper saving:** Just like in the previous case, by using this application to create notes and exams there is no need for paper, something that benefits the environment and the educative expenses, which become lesser for both families and centers.
- **Digital studies:** It allows students to carry their notes to any place, as long as they have a Smartphone. This prevents students from having to carry books and sheets and eliminates the possibility of forgetting part of their study material. Besides, being able to store all the notes in a digital format is good for the students' health, because they don't longer need to carry heavy books.

## 4 STATE OF THE ART

Since the software crisis of 1970, software development is full of problems (Dijkstra, 1972). The Model Driven Engineering appeared due to the existence of this problems and the need to solve them (Kent, 2002). With the creation of the MDE and the models that can be generated through it as a reference, the Domain Specific Languages were born. This helped to simplify the creation of high level languages that were used to improve and manage the software development problems in an easy way, thanks to the power of expression that this languages grant when trying to solve a problem from a certain specific domain (Arie Van Deursen et al., 2000). By using MDE and DSLs we achieve several improvements in the software's development. Among these is the portability, something very important for the development of applications for multiplatform mobile devices, because is not viable having to duplicate the portable elements due to the chances of errors and the loss of time derived from this. In this research, the elements are the text, the images, the sounds and the videos, all of them common to all platforms. In order to improve the development of multiplatform educative videogames it is essential to offer as much simplicity as possible, reducing the amount of money and time spent in the process, as well as the number of problems.

### 4.1 *Development of applications for mobile devices*

Nowadays there is a wide range of electronic devices in the student's daily life. This offers endless possibilities, as it allows to do the exercises and learn by playing anywhere, as long as they have a Smartphone, a laptop, a tablet or any other similar devices, even in places with few technological means or where the access to the internet is not possible (Kam & Rudraraju, 2007). But, because of the great variety of electronic devices it becomes very difficult to develop applications for all of them efficiently (Lavin-Mera et al., 2007). This is due to the big differences that exist between the different mobile operative systems in the market. For instance, Android uses Java, iPhone uses Objective-C and Windows Phone uses C#, and then

there are Firefox OS and Ubuntu for Phones, which will be arriving soon to the market. As it can be seen, evolution in the mobile world is fast and constant.

#### 4.2 *Videogames in education*

The learning through videogames is nothing new. For instance, with Age of Mythology, people learned about various mythologies from the ancient world and Age of Empires contained information about ancient civilizations, their heroes, their weapons, their wars and their history (J. P. Gee, 2003). With Civilization IV it was proven that students from 10<sup>th</sup>, 11<sup>th</sup> y 12<sup>th</sup> grade learned about the history of civilizations by giving life to all kind of ancient empires (Moshirnia, 2007). There are also economy games, like Patrician and Pharaoh, in which the player have to manage an empire, check the needs of the citizens and raise money, something that helps them to understand better the way economy worked back then, as well as experiencing a whole history lesson while playing an entertaining game. Other games of the same style, like SimCity 2000 are useful to learn about urban geography, as (Tüzün et al., 2009) shows.

On the other hand, other games depend on the player to make some moral decisions on the entrusted requests in order to advance through the game, like what happens in Star Wars: Knights of the Old Republic (J. P. Gee, 2003). There are other videogames that allow the user to create its own map for the game, with the possibility of generate things and not only learning, which is what usually happens in schools, as (Brown, 1994) says. This way, the videogames are able to motivate the students and encourage them to solve problems, reflect and think in a more entertaining way (J. Gee, 2005). This makes students learn through participation instead of memorization (J. Gee, 2005). There are researches that show that videogame players improve their reaction times, and also their hand-eye coordination and self-steem (Griffiths, 2002).

#### 4.3 *Model Driven Engineering*

Model Driven Engineering popped up in the last years (MDE) (Kent, 2002) as a motivation to solve certain problems in the development of software, such as: the low quality of the developed software, the unfulfillment of the planning, budget and maintenance cost as the project gets bigger. These problems have been occurring for many years in software development projects (Dijkstra, 1972; González et al., 2008). One way of solving this problems is to automate certain processes of software development to deal with the complexity of its design and implementation in order to obtain a much more reliable software, with more

sophisticated functionalities (Selic, 2008). By using Model Driven Engineering we manage to raise the abstraction level over the normal languages through models so we can use concepts that are close to the domain of the problem. This makes easier to create a Domain Specific Language that represents the specific problem and helps to solve it. The creation and use of a new language in a superior abstraction layer has repercussions on the productivity, raising it (Selic, 2008).

By applying that engineering, we manage to simplify the abstraction level for the different generated videogames. This gives us the necessary abstraction to create the Domain Specific Language that is required to make easier to create games and port them between platforms, one of the main objectives of this research.

#### 4.4 *Domain Specific Languages*

A Domain Specific Language (DSL) is a language that allows to solve a certain specific problem within a specific domain, generally in a declarative way, which usually becomes a call for sub-processes. Because of this, its main characteristic is that it can be expressive (Arie Van Deursen et al., 2000). Using a DSL offers certain advantages. Among them, we should highlight the high improvement of productivity, its easy maintenance and its reliability, as its very unusual for errors to pop up (Reed, Halpern, & Starr, 1996). Its portability, the knowledge of the domain that it offers and the possibility of re-using it for different purposes are other interesting features (A Van Deursen, 1997; Arie Van Deursen et al., 2000). However, its efficiency is worse than the efficiency of native codification and makes harder to find and create the domain (Arie Van Deursen et al., 2000; Reed et al., 1996).

And so, with the creation of our Domain Specific Language we simplify things both for the users and the programmers. With this we make possible for any person to define the desired videogame within our domain in a quick and simple way, creating a supported language for the different platforms, which guarantees the re-usability of all the elements they share.

#### 4.5 *Videogame Editors*

Currently, there are several tools that make easier to develop videogames. Some of them allow to export the game to various platforms at the same time. Thanks to this, the development speed is very high, because you only have to develop the game once and the tool creates the videogame in native code for different operative systems, like Android, iOS, PlayStation, Xbox, etc. Among these, some allow the creation of 2D and 3D games. In the end, the biggest differences between the different editors are: the type of games supported (graphic adventure,

Anexo

---

simulators, arcades, etc.), the possibilities for editing and expanding existing videogames, the usability of the tool itself and the straightforwardness that offers when creating a videogame. In this section we will describe some of the editors used by fellows from other research projects:

- **GameMaker Studio** is a tool for developing multiplatform videogames quickly (YoYoGames, 2013). It's designed to be used by inexpert users, as the test performed with students from elementary school for the research of (Baytak & Land, 2010) shows. This editor allows to create the levels and menus of the videogame in a graphic way, by dragging the elements where we want. It also allows to program in its own language, GameMaker Language, to expand and customize the games, which requires some basic programming knowledge at least. Once the game is created, the editor generates an executable file, which doesn't allow further modifications without using the tool.
- **Unity** is a videogame engine developed by Unity Technologies (Unity Technologies, 2013). It allows the creation of 2D and 3D multiplatform videogames. It features a graphic interface that allows to drag components and customize them with the editor. This allows to modify the physics, graphics, behavior and interactions of the elements. It also allows to import object models created with graphic design tools. It also has its own scripting language to program actions in videogames, adding behaviors and making them more customizable (Mattingly et al., 2012). However, once the game is generated, the editor gives a compiled solution, which makes impossible to edit the game or its levels without using the editor.
- **eAdventure** is a project research created by (Lavín-Mera et al., 2008; Lavín-Mera et al., 2007; Marchiori, del Blanco, Torrente, & Fernandez-Manjon, 2012) to simplify the creation of educative videogames. With eAdventure the development cost for educative videogames is drastically reduced and new educative characteristics are added to the basic contents of videogame development software. It only supports graphic adventure-like videogames, allowing to customize the menus and interfaces (E-UCM, 2013). These games are based on scenes containing mini-games, and motion graphics and videos are used to create transitions between those scenes (Marchiori et al., 2012).
- **Gade4All** is a national project research financed by the AVANZA plan and performed by the Model-Driven Engineering Research Group (MDE-RG) of Oviedo University, which resulted in the creation of a 2D native multiplatform videogame editor. Model Driven Engineering was used to make easier to port the games between platforms. One of the main pillars of this platform is the creation of a Domain Specific Language (DSL) that allows to define the main characteristics and working of the games' taxonomies in a specific and non platform-dependent way. Using the Domain Specific Language, a set of predefined templates and a transformation process, videogames are generated for distinct platform in an automatic, simultaneous form and with a minimum effort. Among the possible types of videogames that can be generated there is the Trivial typology, which has been used to create the educative videogames used for this research, whose functioning we will explain through the proposal. This editor allows the creation of videogames through a graphic interface, with their levels and menus, which can be viewed as they're being created. It also lets you try them by using the HTML5 template in the web, which pops up after selecting the "Try" option. Once the game is generated, a folder (which is fully editable) with all its font code is created before the compilation process.

## 5 CASE STUDY

In order to solve the aforementioned problems, several educative videogames were developed by using the tool for the Gade4All project research. This tool contains an editor which can create videogames through templates. In this project, the Trivial template is used as a base for creating educative videogames. With this editor we created different types of educative videogames centered in different fields: science, history, mathematics, geography, general culture, languages, etc. In this case, the editor has its own image library to create this kind of games. However, these images can be modified and new ones can be added to the library, which allows the customization of the created games. All the educative videogames generated for this case study first offer an educational part and then ask questions about its content. However, this can be manipulated at will by placing the information and the questions in the order we prefer the most. After getting to the end of the game, the player immediately obtains his score. All the generated educative applications run in any mobile device (Android, iPhone y Windows Phone) and also in any computer with a web server containing the educative game in HTML5 format.

### 5.1 *Application*

Educative videogames can be generated through the use of a graphic editor or by editing the XML files. This way is possible to configure the different menus of the generated game. For instance, it's possible to change the main menu, the options menu, the level select screen, the score screen, etc. Once the menus are created it's possible to generate the different levels, in which we can add all kinds of questions and as many as we want. All the questions share common elements and any kind of question can have multiple answers, include sounds and videos and add properties from other type of questions. There is also a time meter that can be adjusted individually for each question in order to establish a time limit.

When the game is generated, we simply choose the platforms we want to export it to: HTML5, Android, iPhone y/o Windows Phone. All of this is performed without writing a single line of code. However, the generated project could be modified and re-compiled when desired, because it's stored in a separate, editable folder. This way, people with technical knowledge could always make modifications whenever they want to change something.

#### 5.1.1 *Types of questions*

In this section we will describe, in general terms and with a high level of abstraction, the possible kind of questions that can be created. Figure 1 shows an example of these, which can be divided in six groups: tutorials, mathematics, trivial, mental agility, interaction and software.

However, any type of action and customization for a question is applicable to any other. For instance, every question can contain video and audio and the actions performed in the interactive questions or the mental agility ones can be performed even in the tutorial question. Still, these six groups are merely orientative, because the customization possibilities allow the user to create new kinds of questions:

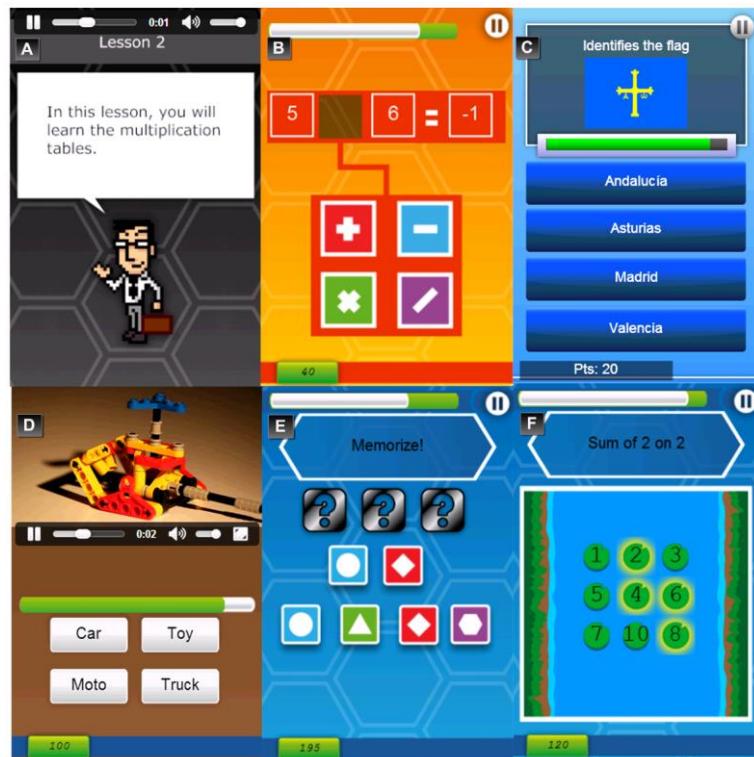


Figure 7 Collage featuring examples of the different types of questions.

#### 5.1.1.1 Tutorials

As it can be seen in Figure 1A, the teacher can include the information that he wants to give to the student. Like the normal questions, this is totally customizable and multimedia components can be added, like video and audio files. This way, the teacher will be able to teach the student by creating one or more tutorials, followed by an educative videogame.

If implemented, the audio and video files can be played, paused and stopped at will by the student.

In **!Error! No se encuentra el origen de la referencia.** we can observe that, aside from the explicative text, there is a control for the volume. In this case, it's possible to offer an audio tutorial to make things easier for the student and let him choose between reading and listening.

#### 5.1.1.2 Mathematics

As seen in Figure 1B, with the use of the mathematic questions the teacher can help the student to practice and improve his calculation capacity and mental agility. In these, the answers can be a number, an operational symbol or both, if we choose to create a multi-answer question. For the afore mentioned, then there's the option of creating more complex operations by adding brackets and square brackets. This way, there can be many possible right answers that have to be selected. In the given example the answer is an operational sign and it's up to the student to determine which one is the correct choice.

#### 5.1.1.3 Trivial

**¡Error! No se encuentra el origen de la referencia.C** and **¡Error! No se encuentra el origen de la referencia.D** show examples of Trivial-like questions. One features an image and the other a video. This kind of questions can contain both images, texts, videos and sounds, so the teacher is given a lot of freedom to create his own questions. For instance, the teacher can include images of pictures and drawings in his questions about history and art, or images of clocks to ask the children what hour is displayed on them, maps and flags of countries for the geography lessons or even sound files for the questions about music. Regarding the text of the question, the teacher can write what he wants, even numbers and mathematical signs, which are useful for this kind of questions. Another possibility would be to include videos for the creation of cultural games about movies, documentaries, history, theater, interviews or videos of experiments performed in the physic and chemistry lessons, even videos of exercises for physical education lessons.

**¡Error! No se encuentra el origen de la referencia.C** features the Asturias flag and the student is asked about the province that this flag represents, so he must identify the flag through visual recognition and choose where it belongs from four possible options, where only one is correct.

In **¡Error! No se encuentra el origen de la referencia.D** the student must identify the object that is being shown in the video, a toy. For this purpose, the student can play the video, pause it or choose an answer.

#### 5.1.1.4 Mental Agility

With this tool, the teacher can configure the elements on the screen so they disappear after a small amount of time. Thanks to this, he can create questions that allow the student to test his mental agility. Figure 1E is a good example of these questions, as it features a memory question,

where three elements are shown, later they are replaced by question marks and a second set of images appear, showing only two of the three initial elements. Seconds later, the four seconds answers appear and the student must choose the element of the first set of images that is missing in the second one.

Other examples of this kind of question could be: give the image of an element and asking the student to find it in a collage with similar image, count the number of times that an element appears in an image or search the differences between two very similar images.

This way, the student would exercise his cognitive capacity in an entertaining way, while relaxing without losing his concentration.

#### **5.1.1.5 Interaction**

Teachers can also create interactive questions. As it can be seen in Figure 1F, it's possible to create a sequence of additions so the student selects the proper way to reach the last number, which will act as a goal. Another example of interactive question could be a geographic map in which the student would have to search for a certain country or province. This could be also applied to other subjects by changing the elements to be found.

With this kind of functionality, the student is motivated to perform a task that would normally be boring in a more entertaining way, because he would have to interact with the game in a particular way, trying to step into the right stones without falling into the water.

#### **5.1.2 Software**

To create the templates for the educative videogames in each platform, we used the proper environment for each case. To create the game template for Android mobiles, we used Eclipse's Software Development Kit (SDK) for Android. To create the game template for iPhone, we used xCode and a Mac OS X operative system. For the development of the game template for Windows Phone we used Visual Studio 2010 and a Windows 7 operative system. For the HTML5 template, we simply used a text editor.

To deploy the HTML5, we used the latest version of the Apache web server. To check its proper operation and perform the tests several internet browsers were used: Chrome, Internet Explorer, Firefox, Opera, Safari y Android, etc. The tests included both the desktop and mobile versions.

The Smartphones used to test the Android version were different in size and worked with different versions of the Android operative system: 2.2, 2.3, 4.0, 4.1 and 4.2, as well as different

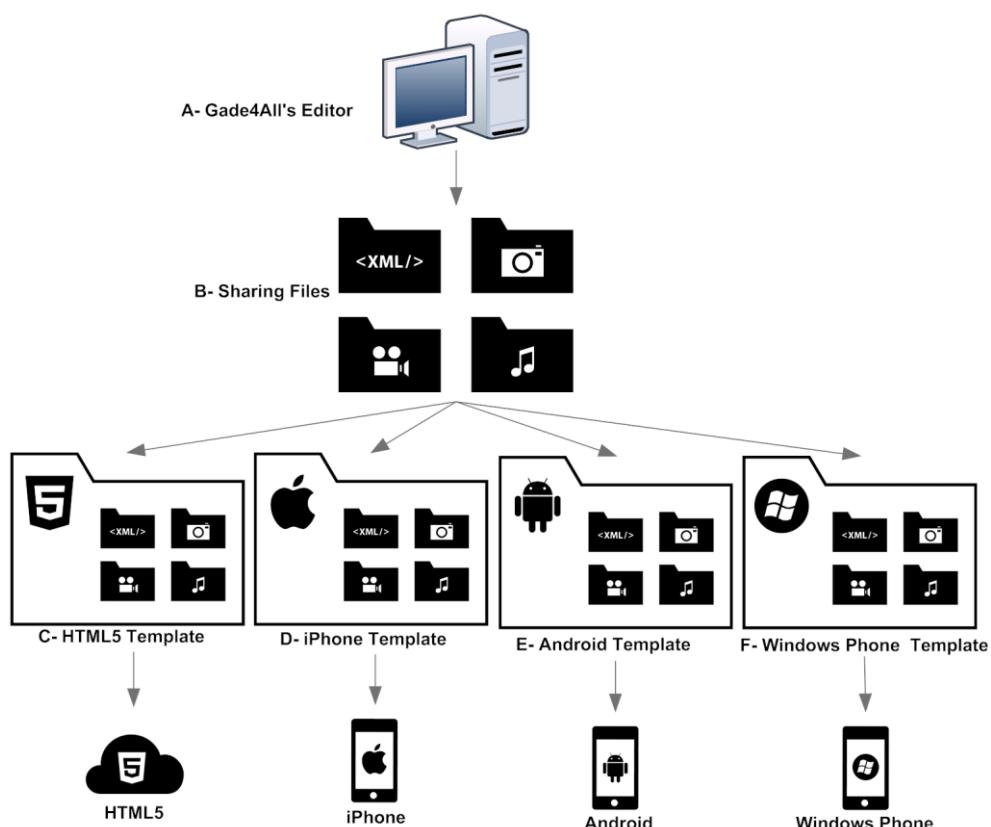
configurations for the emulator. The iPhone version was tested both in the emulator and the Smartphones with the 5 and 6 versions of the operative system. The Windows Phone version was tested on the mobile devices with the 7, 7.8 and 8 versions.

Once the test process was completed, different educative games were generated for all of the supported platforms by using the Gade4All Project's editor.

## 5.2 *Proposed architecture*

The system's architecture is based in a desktop editor that contains the game templates in the different platforms for the different devices. When creating the educative videogame, the editor transforms the templates that we wish to generate and adds them some changes, depending on the options that we choose.

It must be noted that all the XML can be manually created and inserted in the proper folders or we can use an editor to generate them. In this case, we used the Gade4All project's editor to generate the XML and the educative videogames for each platform in a quick and simple way.



**Figure 8 Deployment architecture of educative videogames.**

Figure 8A shows the computer in which the games are generated by using the Gade4All project's editor. This way, the same educative videogame can be generated for each of the four supported platforms at the same time: HTML5, iPhone, Android y Windows Phone.

Once we define the in game screens, the editor (Figure 2B) generates the XML of the menus and adds the images and sound used in these to a temporal folder, following the final diagram. After this, the levels of the game and the questions contained in each one are defined. When all of the levels and questions are defined, the program generates a XML file for each level and all the images, sounds and videos used in the previous folder are inserted.

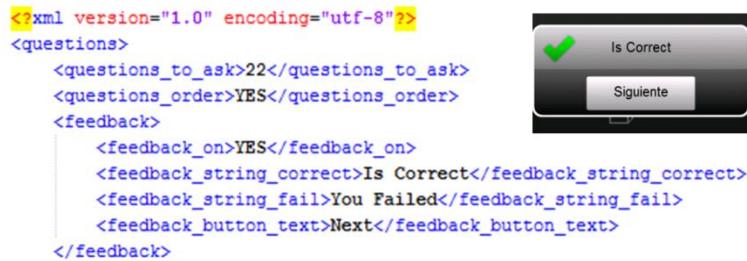
After that, we select the platforms in which we want to release the educative videogame. In this case, we choose the four supported platforms. Then, the editor takes the templates (Figures 2C, 2D, 2E and 2F) containing all the videogame's code, transforming and copying them in the paths specified in the editor in order to insert them in the temporal folder which contains the XML of the menus, levels, images, sounds, videos, etc. After this process, the educative game is created and ready to be played.

### 5.3 *Implementation of the architecture*

The architecture of the system can be divided in two big groups. The first one is the Domain Specific Language created to abstract the code that is common to all platforms. This one contains all the information about the menus and levels of the application and it's the same in all the platforms. This way, we generate a language that can be interpreted by the native language of each device, which implies a smaller data redundancy and also simplifies the creation of ubiquitous educative videogames. The second group are the templates, which contain the native font code of each platform but not the images nor the information about menus and levels, which means that they're not compilables.

#### 5.3.1 Domain Specific Language

To make the process of porting the application between platforms easier, we used Model Driven Engineering. This way, we created a DSL divided in two eXtensible Markup Language (XML). These XML are files that contain the information of the menus and levels of the educative games. There is a XML for each level and a XML with the information of the different menu screens. The level files are numbered starting from L0 and increasing in one with each new level. The images that represent each level are numbered this way and they are shown in the level selection screen. This XML are common to each platform, so the same level/menu file can be used in any platform. The XML are composed by three parts:



**Figure 9 Example of DSL that contains the tags that define the feedback for an answer.**

Figure 9 shows the first part, the parent node: **questions**. Within this node all the questions are stored under the **question** tag and the information about the quantity, order and feedback parameters shown when answering a question. This node also houses data that is common to all the questions of that level. These are the **question\_to\_ask**, **questions\_order** tags and the child node **feedback**.

The **question\_to\_ask** tag indicates the total number of questions the level has. In this number are included the tutorial screens and it can show an inferior number to the total amount of questions included, if we don't want to show all the questions included in the aforementioned level.

The next tag is called **questions\_order**. This tag indicates if the questions that we will be asked will be shown in the same order that is specified in the XML (*YES*) or if they are shuffled every time we play the level (*NO*). If the **question\_to\_ask** tag does not contain the total number of existing questions in the XML file, we can create exam levels with a great amount of questions presented in order. This way, we can create different exams and tests for each student using the generated educative game.

The children node **feedback** shows the information that will be shown when answering a question, if desired. This one is composed of four tags that configure the feedback: **feedback\_on** can be activated or deactivated after answering a question. With **feedback\_string\_correct** we specify the chain of text that we want to display when a question is answered correctly. With **feedback\_string\_fail** we specify the chain of text that we want to display when we choose a wrong answer. For the text in the button we use the **feedback\_button\_text** tag. In Figure 9 we can see the simple code for this node and, in the right part, the graphic example corresponding to a correct answer:



Figure 10 *Example of DSL containing the tags that define a time meter and a scoreboard.*

Figure 10 shows an example of the first part of the **question** node and the graphic example corresponding to the code shown in the status bar and the scoreboard. This one is composed of three subsections: The children of the **question** node, the **status\_bar** children and the **scoreboard** children. **Question** has three tags that are used to define the time limit for the question (**question\_time**), the points received for a correct answer (**question\_points**) and if it is a tutorial or a normal question (**question\_is\_description**).

If we want to create a tutorial screen (for example, an explanation of the multiplication tables) the **question\_is\_description** tag must contain *YES*. In that case, points and time will not be taken into consideration.

The second part is the **status\_bar** children. This node defines the aspect and position of that bar, which is composed by two images, the back image (**status\_bar\_image\_background\_source**) and the front image, which is the one that becomes smaller as time passes by (**status\_bar\_image\_front\_source**). With the other tags define the central position of the image in the x axis (**status\_bar\_x\_pos**) and the y axis (**status\_bar\_y\_pos**), as well as the width (**status\_bar\_width**) and weight (**status\_bar\_height**) of both images.

In the third part is the **scoreboard** node. This defines the image used for the scoreboard (**scoreboard\_image\_source**), its central position in the x axis (**scoreboard\_x\_pos**) and the y

axis (**scoreboard\_y\_pos**), as well as its width (**scoreboard\_width**) and height (**scoreboard\_height**). This node also has tags that define the font, family, style, color, size of the text, as well as its horizontal alignment regarding the image that contains it.

```
<graphics>
  <graphic>
    <graphic_is_collided>NO</graphic_is_collided>
    <graphic_is_correct></graphic_is_correct>
    <graphic_order></graphic_order>
    <graphic_image_source>question.png</graphic_image_source>
    <graphic_image_pressed_source></graphic_image_pressed_source>
    <graphic_font_generic_family>sans-serif</graphic_font_generic_family>
    <graphic_font_font_family>times new roman</graphic_font_font_family>
    <graphic_font_style>italic</graphic_font_style>
    <graphic_font_size>20</graphic_font_size>
    <graphic_font_color>white</graphic_font_color>
    <graphic_x_pos>160</graphic_x_pos>
    <graphic_y_pos>78</graphic_y_pos>
    <graphic_width>278</graphic_width>
    <graphic_height>86</graphic_height>
    <graphic_text>What is the capital of Spain?</graphic_text>
    <graphic_text_align>center</graphic_text_align>
    <graphic_time_to_appear>0</graphic_time_to_appear>
    <graphic_time_to_desappear>0</graphic_time_to_desappear>
    <graphic_audio_source></graphic_audio_source>
    <graphic_audio_autoplay></graphic_audio_autoplay>
    <graphic_audio_loop></graphic_audio_loop>
    <graphic_audio_controls></graphic_audio_controls>
    <graphic_video_source></graphic_video_source>
    <graphic_video_autoplay></graphic_video_autoplay>
    <graphic_video_loop></graphic_video_loop>
    <graphic_video_controls></graphic_video_controls>
  </graphic>
```

**Figure 11 Example of DSL containing the tags that define a graphic belonging to a question.**

Figure 11 shows the code of the second part of the **question** node. The rest of the **question** node is formed from the **graphics** node. Within it, we must place the **graphic** nodes in the order they're painted. The first to appear will be the first to be painted, so if the second is painted over the first, it will be screened. Each graphic node is a new layer, as it happens in any of the most common graphic edition softwares.

This type of nodes contains all the required information to paint a graphic in the screen. Grouping them results in the creation of a question. The great amount of tags it has and the variety of them, makes this node the most complex of all. This is because in the game everything is a graphic: backgrounds, buttons, images, sounds, videos, etc. This way we have

Anexo

---

the required power to put all the workload in the auto-generated part that is contained in the templates (the logic), making the creation of games easier and more intuitive.

The first thing to do is pointing out if the graphic is collidable (**graphic\_is\_collided**). This way we can decide which elements we can interact with, as well as creating elements that change their image when they're clicked. To distinguish a correct answer from a wrong one we use the **graphic\_is\_correct** tag.

With the **graphic\_order** tag we can define the order in which the graphics must be clicked. For instance, if we display a set of numbers and we want the student to click them in certain order, we must assign a number to each graphic to define that order. If we don't want to establish any order, we simply leave this field empty.

To assign an image to a graphic, we have to put its name in the **graphic\_image\_source** tag. If we want this image to change after a while or when the graphic is selected, we have to place that new image in the **graphic\_image\_pressed\_source** tag. As with other tags, the central position of the image regarding the x and y axes, its width and height are defined in the corresponding tags: **graphic\_x\_pos**, **graphic\_y\_pos**, **graphic\_width** y **graphic\_height**.

To define the text of the graphic it must be written in the **graphic\_text** tag. The text will be aligned to the vertical center of the image by default. This tag supports carriage returns, so it's possible to leave blank spaces to fill them with images or more text. As the previous nodes, here we can configure the text's size, Font, style, color, alignment, etc. With the **graphic\_time\_to\_appear** and **graphic\_time\_to\_desappear** tags, we can make the graphic appear or disappear after some time or even both. This way, we can create questions where a graphic appears after X seconds while others disappear. In the given example some graphics appear by default in the second 0 and after a few seconds they disappear while others appear.

Lastly, we have the tags that define the sound and the video, which require four tags to control all the options. First, we change the name of the file with the **source** tag. The file can play automatically as the question starts (**autoplay**), be restarted when it ends (**loop**) or show its audio/video controls (**controls**).

Below we show the XML of the different screens:

Anexo



The screenshot shows an XML code editor with a hierarchical tree view on the left. The root node contains global settings like the number of levels (7), screen width (320), screen height (480), orientation (portrait), and a main menu view. The main menu view includes details for a start button, such as its image (button.png), alignment (center), size (20x20), font (courier new), color (white), and coordinates (160, 270). Below the main menu, there are collapsed sections for music, pause, select level, play again, end game, images levels, and loading views.

```
<number_of_levels>7</number_of_levels>
<screen_width>320</screen_width>
<screen_height>480</screen_height>
<screen_orientation>portrait</screen_orientation>
<menu_view>
    <image_start_button>button.png</image_start_button>
    <start_button_align>center</start_button_align>
    <start_button_size>20</start_button_size>
    <start_button_generic_font>courier new</start_button_generic_font>
    <start_button_color>white</start_button_color>
    <height_start_button>78</height_start_button>
    <width_start_button>230</width_start_button>
    <pos_x_button_start_menu>160</pos_x_button_start_menu>
    <pos_y_button_start_menu>270</pos_y_button_start_menu>
    <start_button_text>Play</start_button_text>
    ...
    ...
</menu_view>
<music_view>
<pause_view>
<select_level_view>
<play_again_view>
<end_game_view>
<images_levels>
<loading_view>
```

**Figure 12 XML containing the tags that define the menus and levels used in the game.**

In Figure 12 we can see the menus' XML. This XML contains all the information about the different screens and the game's global characteristics: the information about the total number of levels of the educative game (**number\_of\_levels**) and the width (**screen\_width**) and height (**screen\_height**) of the screen in which it was generated. This is because, aside from being multiplatform, the generated educative games fit to the screen of any device. That's why it's necessary to indicate the width and height of the screen for the correct functioning of the resolution adapter. With the **screen\_orientation** tag the game changes the visualization of the screen from vertical to horizontal and vice versa.

The rest of the XML is composed of two children nodes. Each node is a screen of the game. The main menu (**menu\_view**) is where the student chooses between playing the game, changing the options or exiting the game. The pause screen (**pause\_view**), as its name suggests, is for pausing the game in the middle of a question. The level selection screen (**select\_level\_view**) allows the student to select which level he wants to play. The “play again” screen (**play\_again\_view**) appears after checking the score for a level and lets you choose between replaying the level or going back to the level selection screen. The screen that appears after a level is completed (**end\_game\_view**) shows the score for that level. The loading screen

(**loading\_view**) appears while the game is loading. There is also an **images\_levels** node, which contains the images that represent each level in the level selection screen. Each node contains all the customizable elements of the screen it belongs to. For instance, **menu\_view** contains the image of the *Play* button, as well as the text of that button, with all its characteristics. The same happens with the rest of elements: *Options* button, *Exit* button and the background image. The remaining images follow the same standard: each element on the screen it's read from the XML, because every element is customizable thanks to the templates of each platform.

### 5.3.2 *Templates*

The templates are created through an educative videogame natively created in each platform: Android, iPhone y Windows Phone. The only difference with that native videogame is the absence of XMLs, images, videos and (in the Android version) a compilable R file. At the beginning of the project, we studied the four systems to see the possibilities that each one offered. After this, we started creating the first template for the most restrictive system. Then, we ported it to the other system, keeping the names, the folder structure, the classes, the objects and the attributes, adequating everything to the different programming languages and the possibilities of the systems to paint images, display text and play audio and video. This way, to convert the templates for the desired system, we only have to insert the XMLs, images, sounds and videos in their folders and then compile everything.

## 6 EXPERIMENT

This section is divided into two sub-sections: methodology and prototype. The first one describes the methodology used for the Gade4All national project research and the performed tests. The second one describes the editor used for the tests and the way it generates the Domain Specific Language.

### 6.1 *Methodology*

To test the performed research we followed this procedure: First, we created the educative videogame in native code for each platform. After testing everything we created the templates and started creating the Gade4All project editor to be able to generate the games in a quick and simple way. When the editor was completed, we performed exhaustive tests to check the proper functioning of all the functionalities that we wanted to offer when creating ubiquitous games.

There were two major tests: The first one was taken by students of Computing Science from Oviedo's University and the objective was to solve some of the editor's usability problems, as well as problems with the generation of the educative videogames or problems with the games

itself. Through this test we sought to improve the editor and the templates for the second test and also fixing some errors in the games.

The second test also took place in the Oviedo's University as a contest open to every student and teacher from the faculty, in which the contestants had to create a videogame using the Gade4All editor.

During the contests we took good note of the errors that popped up and the things that weren't very clear for the users when creating their games. Each contestant had a personal assistant for the test, which made easier to locate any error in the tool.

## 6.2 *Prototype: Gade4All Trivial*

In this section we will explain how questions are created with the latest version of the Gade4All project's editor and its connection with the DSL. After that, we will explain one of the applications created for the contest.

### 6.2.1 Comparison between the editor and the DSL

In this section, it is explained the equivalence between the graphic editor and the DSL generated by the tool. For this, is used a fragment of the domain specific language generated by the transformation motor.

Anexo

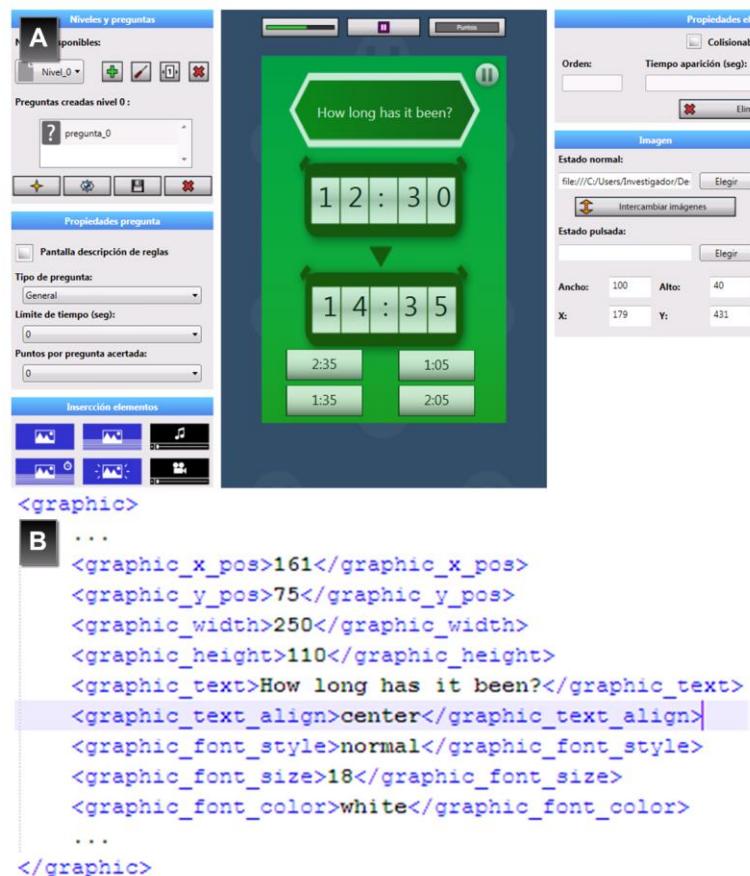


Figure 13 Comparison between a question created with the Gade4All editor and a sample of the generated DSL's code.

- Graphic editor:** Figure 13A shows the creation of a question with the editor. Here we can choose a default question and modify it adding new elements, which are fully configurable. In this example we created a question about time and we modified it to include images of timers, change the color, font, size and style of the text and the possible answers. After defining it and saving the changes, we go to the code generation screen. Here, we select the platforms in which we want to publish the educative game, which will be exactly the same in every platform.
- DSL:** Figure 13B shows a part of the code of the DSL generated by the question created in the editor, which is common to all the platforms. The example shows part of the code of the *graphic* node of the image in which the text of the question is inserted. Here we can see its position on the screen, its dimensions and its properties.
- Transformation engine:** The editor's transformation engine converts the templates of the chosen platforms for which the game will be generated. After that, the images, sounds, videos and XML files are inserted in the folders of each template's transformation. It must be noted that this last step inserts the same contents in all versions, because the XML uses the created Domain Specific Language and that allows us to make ports to all the different platforms. And that's how the game is generated.

### 6.2.2 Educative Videogame created in the contest

Figure 14 shows six screenshots of an educative videogame about the multiplication tables, which was created by a student in the contest. In Figure 14A there is an explanation of the lesson, followed by a multiplication table in Figure 14B. After watching all the multiplication tables, the student is asked several questions in which he will have to choose the correct answer, as it's shown in Figure 8C and Figure 8E. The creator of the game chooses to include feedback and Figure 8D shows the feedback created when the student selects the right answer. Once the level is finished, the player of this educative game can check his score immediately and see how many questions he has answered correctly and how many he has failed, as shown in Figure 8F.

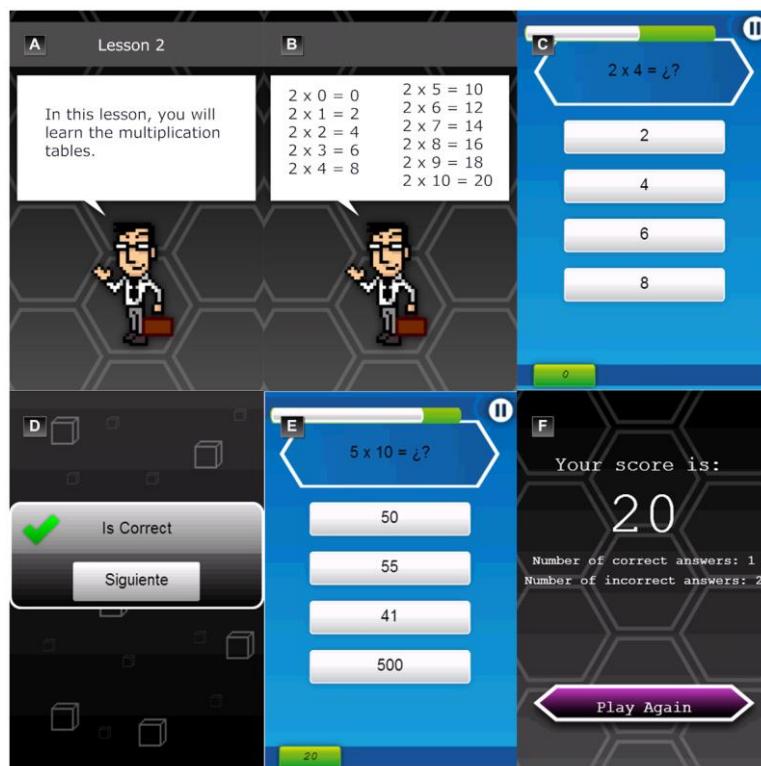


Figure 14 Example of an educative game

## 7 EVALUATION RESULTS

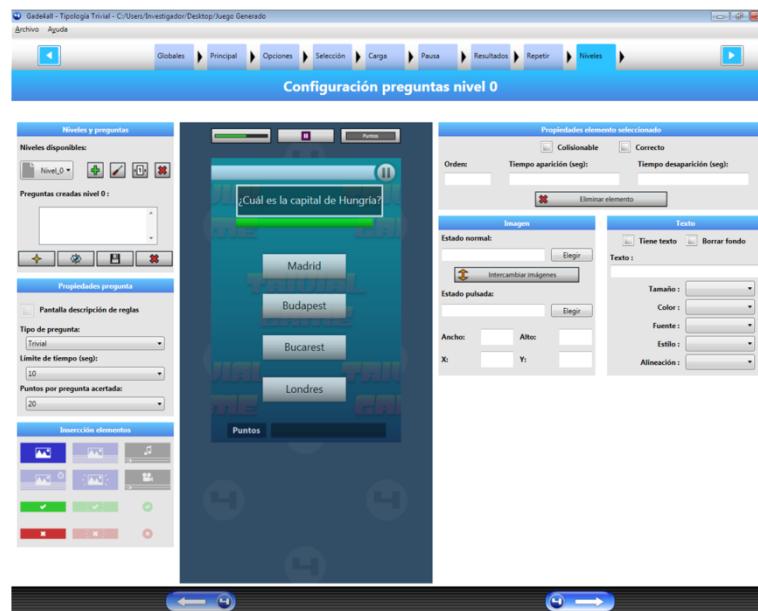
After finishing the second test at the contest, the data obtained by the contestants was evaluated. After the evaluation, Gade4All editor was modified in order to correct the errors and make it more usable.

Anexo



**Figure 15 Image with the first version of the Gade4All editor**

Figure 15 shows screenshots of the menus where the questions for videogames of the Trivial typology are created, two from the editor that was used in the contest (Figure 15A and Figure 9B) and one from the improved version (Figure 16C). As it can be seen, the improved version's interface is much more usable and easy to use by inexpert users.



**Figure 16 Image with the new version of the Gade4All, created after retrieving and evaluating the feedback from the users.**

The main change affected the contents shown in the screen. In the old version there were two menus, one for the levels and the questions contained in them (Figure 9A) and other to edit each question (Figure 9B). In the second one we can see that there was an image for each kind of question and big images to access the configuration menus of other elements, like the time

meter or the scoreboard. Besides, the icons and texts weren't as representative as it was thought. This created confusion among the users and that confusion lead to mistakes, making the videogame creation process slower, tedious and even incorrect, because some of them weren't able to do what they intended to do.

The new version (Figure 15C) merged both menus in a much more simple one. The first menu was changed into a multiline text field with multiple selections along with a drop-down to avoid any loss of functionality while keeping the interface usable. The images where changed for others smaller and more descriptive, texts were rewritten, more significant and extensive tooltips were added, as well as a bigger number of drop-down menus to prevent an overload of visual elements in the screen and reduce the amount of information shown in the screen.

After the changes, the latest version of the editor was tested by the vast majority of users that performed the previous tests, which were satisfied with the new interface and were able to use the editor more fluently. The previous experience with the editor helped, of course, but the changes in the man-machine interface resulted in a significant improvement on the usability of the application.

## 8 DISCUSSION

As it can be seen throughout this proposal, we managed to create a tool for generating educative videogames in a simple way: the Gade4All editor. The editor was considerably improved thanks to the two tests performed on it, which helped to locate several errors that were fixed after the analysis.

However, this type of games can be created with any other editor as long as it implements these templates and the created Domain Specific Language, that is to say, an editor with a similar transformation engine. They can be also created manually by editing the XML that contain the Domain Specific Language and insert them in the template's folder along with the images, sounds and videos that are used. In our case, we managed to improve the usability of the Gade4All editor through the two tests performed, making it easier to use, which was the main purpose of this project research.

Although the generated educative videogames might look simple for being in 2D, it must be noted that the vast majority of mobile videogames are, in fact, bidimensional. This makes easier to use the editor and also simplifies the customization of the videogames. As we will explain in the future work section, it would be interesting to create three dimensional videogames or

games that make use of 3D. However, most of today's mobile devices lack the power to use this kind of technologies. That's why we allow the user to create his own images and load them through the editor, and if the user doesn't want to use custom images, he has a graphic library full of pre-designed images that cover all the possible subjects that are taught in schools.

## 9 FUTURE WORK

In spite of what it has been said in this proposal and the achievement of making possible for any person to create 2D educative videogames, this research has a long road ahead regarding future work. While this project gave support to four different platforms (Android, iOS, Windows Phone y HTML5) we could create templates for lots of others. Another interesting improvement would be to give support for three dimensional games so we could create new type of questions and improve the current ones, adding stereoscopy for a better visualization and a much more attractive appearance for the games. Some of these would require an extension of the Domain Specific Language, which makes them part of the future work:

- **Creating templates for other systems:** An important future work would be to create a bigger number of templates for different systems, like:
  - Different desktop operative systems: Windows, GNU/Linux, Mac OS X. Not everyone has a Smartphone or access to the internet, and a desktop application could be quite versatile and would offer a better image quality, as well as more comfort for the user.
  - Other Smartphones with different operatives systems that are about to come to the market, like Firefox OS and Ubuntu for Phones.
  - Applications for videogame consoles (PS3, XBOX360, Wii, Wii U, NDS, N3DS) or Smart TVs.
- **3D games:** Another interesting idea would be to create games in 3D instead of 2D. This would allow us to create new types of questions and add more possibilities to the existing ones. For example, we could include a 3D object that could be rotated by the student to observe it from any angle: a sculpture, a Rubik Cube, polyhedrons made from blueprints, etc.
- **3D support:** Adding 3D support for the videogames through stereoscopy and/or 3D goggles in devices with this option. This way, the games would be more attractive for the students and we could offer new possibilities for the creation of educative videogames, allowing things like observing models and blueprints from different angles, something very useful for technology and technical drawing lessons.
- **Domain Specific Language extension:** Adding new options to the DSL to be able to generate new types of questions, have a bigger customization capacity and offering more versatility. This would be necessary if we wanted to perform one of the two future Works described before. We would have to include a tag to indicate if a question uses bidimensional or three dimensional graphics, because it would require a different behavior

on behalf of the program, as it happens with stereoscopy, which will also need a default value to indicate its initial depth.

## 10 ACKNOWLEDGMENT

This work was performed by the University of Oviedo under Contract No. MITC-11-TSI-090302-2011-11 of the research project Gade4all. Project co-financed by the Ministry of Industry, Tourism and Commerce under its National Plan for Scientific Research, Development and Technological Innovation.

## 11 CONCLUSIONS

As described throughout the whole proposal, we managed to create a Domain Specific Language to create multiplatform educative games in a simple and efficient way, oriented to people without technical knowledge in the creation of videogames. All by using the Gade4All editor, which gives a great versatility to the game's creator.

With the help of the performed tests we improved the template used for creating videogames, which was the main objective of the Gade4All project. If we were using a different editor or the games were created manually, the process would be quite harder, because (unlike the Gade4All editor) these methods are not specifically designed for creating this kind of videogames.

These videogames help both teachers and students. Their portability and the possibility of adding any type of content make them excellent for the students, who can get notes of quality that they can consult anywhere. Besides, these notes can include exercises to practice and revise the concepts acquired through the use of the application.

Teachers, on the other hand, can create their lessons in the form of an educative videogame and motivate their students to learn with it, because the lack of motivation is one of the main problems of today's education. This way, they will be able to improve their students' education by teaching them using a much more entertaining and pleasant method. Another advantage is the possibility to create tests that can be distributed and corrected in short time, helping both teachers and students.

Finally, from the ecologic point of view, this research proves to be interest because of the amount of paper and ink that is saved and the reduction of pollution that derives from the fact that the chemical products that are used to process them. This has a positive effect on the environment, because these products are replaced by the use of a possible renewable energy that generates enough energy to charge the mobile device.

## 12 REFERENCES

- Baytak, A., & Land, S. M. (2010). A case study of educational game design by kids and for kids. *Procedia - Social and Behavioral Sciences*, 2(2), 5242–5246. doi:10.1016/j.sbspro.2010.03.853
- Brown, A. (1994). The advancement of learning. *Educational researcher*, 23(8), 4–12.
- Deursen, A Van. (1997). Domain-specific languages versus object-oriented frameworks: A financial engineering case study. *Smalltalk and Java in Industry and Academia* ....
- Deursen, Arie Van, Klint, P., & Visser, J. (2000). Domain-specific languages: an annotated bibliography. *ACM Sigplan Notices*.
- Dijkstra, E. (1972). The humble programmer. *Communications of the ACM*, 15, 859–866.
- E-UCM, U. C. de M. (2013). eAdventure. Retrieved from <http://e-adventure.e-ucm.es/>
- Gee, J. (2005). Good video games and good learning. *Phi Kappa Phi Forum*.
- Gee, J. P. (2003). What video games have to teach us about learning and literacy. *Computers in Entertainment*, 1(1), 20. doi:10.1145/950566.950595
- González, E., Fernández, H., & Díaz, V. (2008). General purpose MDE tools. *IJIMAI*, 1.
- Griffiths, M. (2002). The educational benefits of videogames. *Education and Health*, 20(3), 47–51.
- Kam, M., & Rudraraju, V. (2007). Mobile gaming with children in rural India: Contextual factors in the use of game design patterns. *Proceedings of 3rd Digital ....*
- Kent, S. (2002). Model Driven Engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 2335(2), 286–298.
- Lavín-Mera, P., Moreno-Ger, P., & Fernández-Manjón, B. (2008). Development of Educational Videogames in m-Learning Contexts. *2008 Second IEEE International Conference on*

*Digital Game and Intelligent Toy Enhanced Learning*, 44–51.  
doi:10.1109/DIGITEL.2008.21

Lavin-Mera, P., Torrente, J., Moreno-Ger, P., Vallejo Pinto, J. A., & Fernández-Manjón, B. (2007). Mobile Game Development for Multiple Devices in Education. In *International Journal of Emerging Technologies in Learning (iJET)* (Vol. 4, pp. 1–8). Tokyo. doi:10.3991/ijet.v4s2.910

Marchiori, E. J., Del Blanco, A., Torrente, J., & Fernandez-Manjon, B. (2012). A framework to improve evaluation in educational games. In *Global Engineering ...* (pp. 1–8).

Mattingly, W. a., Chang, D., Paris, R., Smith, N., Blevins, J., & Ouyang, M. (2012). Robot design using Unity for computer games and robotic simulations. *2012 17th International Conference on Computer Games (CGAMES)*, 56–59. doi:10.1109/CGames.2012.6314552

Moshirnia, A. (2007). The Educational Potential of Modified Video Games. *Issues in Informing Science and Information Technology*, 4, 511–521.

Ozcelik, E., Cagiltay, N. E., & Ozcelik, N. S. (2013). The effect of uncertainty on learning in game-like environments. *Computers & Education*, 67, 12–20. doi:10.1016/j.compedu.2013.02.009

Papastergiou, M. (2009). Exploring the potential of computer and video games for health and physical education: A literature review. *Computers & Education*, 53(3), 603–622. doi:10.1016/j.compedu.2009.04.001

Reed, A. B., Halpern, V., & Starr, J. E. (1996). Little languages: Little maintenance? doi:10.1016/j.jvs.2012.10.105

Selic, B. (2008). MDA manifestations. *The European Journal for the Informatics Professional*, ..., (June).

She, Z., & Wang, Q. (2011). Innovation in Green Technology Research of Forest Products Trade in the Context of Ecological Civilization. *2011 International Conference on Management and Service Science*, 1–4. doi:10.1109/ICMSS.2011.5998683

Tüzün, H., Yılmaz-Soylu, M., Karakuş, T., İnal, Y., & Kızılıkaya, G. (2009). The effects of computer games on primary school students' achievement and motivation in geography learning. *Computers & Education*, 52(1), 68–77. doi:10.1016/j.compedu.2008.06.008

Unity Technologies. (2013). Unity. Retrieved from <http://unity3d.com/>

Yang, C., & Zhao, H. (2011). Barriers to Green Technology Innovation in Large and Medium-Sized Enterprises. *2011 International Conference of Information Technology, Computer Engineering and Management Sciences*, 175–178. doi:10.1109/ICM.2011.350

YoYoGames. (2013). GameMaker Studio. Retrieved from <http://www.yoyogames.com/gamemaker/studio>