Universidad de Deusto
Deustuko Unibertsitatea
Deusto

Facultad de Ingeniería

Plataforma web y metodología para el desarrollo de sistemas sensibles al contexto basada en la colaboración entre programadores y expertos en el dominio

Programa: INGENIERÍA INFORMÁTICA Y TELECOMUNICACIÓN

David Martín del Canto
Bilbao, enero de 2013

Universidad de Deusto
Deustuko Unibertsitatea
Deusto

Facultad de Ingeniería

Plataforma web y metodología para el desarrollo de sistemas sensibles al contexto basada en la colaboración entre programadores y expertos en el dominio

Programa: INGENIERÍA INFORMÁTICA Y TELECOMUNICACIÓN

Tesis doctoral presentada por Don David Martín del Canto

Dirigida por el Dr. Diego López-de-Ipiña González-de-Artaza
y la Dra. Aurkene Alzua-Sorzabal

El Director La Directora El Doctorando

La mejor forma de predecir el futuro, es inventarlo.

Alan Curtis Kay (1971). Informático estadounidense.

Agradecimientos

En primer lugar, quisiera agradecer a la Dra. Aurkene Alzua-Sorzabal y al Dr. Diego López de Ipiña por la dirección de este trabajo de investigación y por todos los consejos prestados en largas reuniones de tesis. Sin su esfuerzo, este trabajo no hubiera sido posible. Por supuesto, agradecer también a CICtourGUNE, por brindarme la oportunidad de empezar esta carrera de fondo y posibilitar la llegada a esta primera meta volante.

Quisiera dar las gracias también a todos mis compañeros de trabajo que en algún momento han seguido de cerca la evolución de esta tesis doctoral, y en especial, a todos aquellos que han aportado su granito de arena prestándose como voluntarios en el proceso de evaluación. Agradecer especialmente a Ion Gil sus magistrales lecciones con la plataforma SPSS y a Carlos Lamsfus, por contextualizarme en cada una de las etapas de este arduo proceso.

Quisiera agradecer también a todo el personal de *Pervasive Media Studio* (Bristol), y especialmente a Verity Alexander, Clare Reddington y Jon Dovey, por su acogida y por darme la oportunidad de compartir mi trabajo con todos ellos durante tres meses.

Tengo que dar las gracias también a otras muchas personas que en algún momento de este largo proceso, han contribuido al mismo en mayor o menor medida. Marina Abad, por su ayuda en la elaboración de los cuestionarios de evaluación, a Lorea por su buen hacer con los mapas, a Jesús Villanueva por sus útiles consejos de usabilidad entre cañas y nachos, a Frank, Olga, Francisco y compañía por hacer más llevaderos mis meses en tierras lejanas, y a todos los voluntarios que se prestaron a realizar las pruebas de evaluación durante mi estancia investigadora en Bristol.

Finalmente, pero no menos importante, quisiera acordarme también de mi familia (lo siento, nunca supe explicaros qué era esto de "la tesis") y amigos, y mencionar especialmente a Lidia, por su paciencia, comprensión y apoyo en aquellos momentos de máxima dedicación.

Muchas gracias.

Nota de traducción

En el texto de esta tesis aparecen numerosas traducciones al español de términos y definiciones en inglés que han sido realizados por el doctorando.

Con ello, no se busca dar con el término exacto sino ayudar al lector a seguir el discurso, dando coherencia al texto y tratando de respetar, en la medida de lo posible, la idea del autor original aunque en ocasiones se hayan podido perder algunos matices.

De esta manera, se ha optado por traducir la mayoría de los términos manteniendo también los conceptos originales: entre paréntesis y en cursiva en el caso de conceptos, y en notas a pie de página en el caso de las definiciones más relevantes citadas en el texto de este trabajo.

En las figuras que representan los elementos de cada una de las arquitecturas de los entornos de desarrollo de sistemas sensibles al contexto analizadas (figuras 6 a 15), se ha mantenido el idioma original. A su vez, tanto en la Figura 16 donde se muestra el esquema de una de las metodologías de desarrollo de sistemas sensibles al contexto analizadas, como en la Figura 17 donde se muestran las diferentes relaciones basadas en lógica temporal, se ha mantenido el idioma original.

Las diferentes figuras que muestran extractos de la interfaz de la plataforma de desarrollo descrita en el Capítulo 5 se encuentran también en inglés, ya que se ha elegido este idioma como base para la implementación de la misma.

Índice de contenidos

Índice de figuras

Índice de tablas

Índice de gráficos

LISTA DE ABREVIATURAS

BI	*Behavioural Intention*
CASS	*Context-Awareness Sub-Structure*
CMF	*Context Management Framework*
CML	*Context Modelling Language*
CoBra	*Context Broker Architecture*
COP	*Context-oriented Programming*
ECA	*Event Condition Action*
EUD	*End-user Development*
GPS	*Global Positioning System*
HCI	*Human Computer Interaction*
HTTP	*Hypertext Transfer Protocol*
J2ME	*Java to Micro Edition*
JCAF	*Context Awareness Framework*
JSON	*JavaScript Object Notation*
LBS	*Location Based Services*
MIS	*Management Information Systems*
PEOU	*Perceived Ease of Use*
PU	*Perceived Usefulness*

RIA *Rich Internet Applications*

RDF *Resource Description Framework*

SAW *Situation Awareness*

SIG Sistema de Información Geográfica

SOCAM *Service-oriented Middleware for building Context-Aware Services*

TAM *Technology Acceptance Model*

TIC Tecnologías de la Información y Comunicación

URL *Uniform Resource Locator*

UTC *Universal Time Coordinated*

WoT *Web of Things*

XML *eXtensible Markup Language*

XP *Extreme Programming*

RESUMEN

La personalización de servicios en entornos computacionales es esencial para una óptima experiencia de usuario. Esta personalización es todavía más relevante en entornos móviles, donde los usuarios tienen que interactuar con dispositivos de un menor tamaño y donde normalmente requieren una información muy específica en un momento y lugar determinados. Así, es crucial que los servicios puedan ser capaces de procesar información sobre el contexto de los usuarios, con el fin de identificar la situación en la que se encuentran y adaptar su comportamiento a dicha situación.

Sin embargo, este proceso de personalización es complejo, ya que son necesarios conocimientos de varias disciplinas científicas y tecnológicas. Por una parte, los datos tienen que ser obtenidos de fuentes de contexto normalmente distribuidas y heterogéneas. Esta información tiene que ser transformada a un modelo de datos sobre el que poder establecer una serie de condiciones para que el sistema pueda detectar la situación del usuario, adaptando en consecuencia el servicio proporcionado.

A lo largo de la última década, se han propuesto diferentes entornos de programación para simplificar el desarrollo de sistemas sensibles al contexto. Sin embargo, existen todavía diferentes carencias en los mismos. En primer lugar, no todos los entornos son capaces de gestionar de manera automática la movilidad de los usuarios. Esta característica es fundamental, ya que la localización es uno de los parámetros de contexto primarios en este tipo de escenarios móviles. Por otra parte, se ofrecen interfaces de programación de bajo nivel, por lo que solamente pueden participar en el desarrollo usuarios programadores. Esta característica hace que la involucración de personal no técnico en el ciclo de vida del desarrollo del sistema, particularmente expertos en el dominio, sea muy complicado. La participación de este tipo de usuarios puede acelerar y mejorar el proceso de desarrollo, además de mejorar la calidad de los productos resultantes. Este tipo de usuarios son los que mejor conocen las situaciones que tienen que ser identificadas por el sistema y los comportamientos del mismo. El objetivo del presente trabajo de investigación es proporcionar una nueva aproximación para facilitar el desarrollo de sistemas sensibles al contexto, incluyendo en el mismo a los expertos en el dominio en colaboración con los programadores.

En primer lugar, se ha diseñado una metodología denominada *Situation-Driven Development* o Desarrollo Dirigido por Situaciones, con el fin de guiar el proceso de desarrollo. Esta metodología se basa en la identificación y parametrización de las situaciones relevantes que tienen que ser detectadas por el sistema sensible al contexto a implementar. El objetivo de la misma es involucrar al experto en el dominio en el proceso de definición de las situaciones y posterior desarrollo del sistema.

Además, se ha implementado un entorno de desarrollo web denominado *Context Cloud* o Nube de Contexto, donde la información de contexto puede ser gestionada por expertos en el dominio, no necesariamente programadores, sin tener que realizar configuraciones de manera programática. La plataforma se ha diseñado teniendo en cuenta la metodología propuesta. Así, el modelo de datos de contexto, el proceso de obtención de información de contexto y las condiciones para identificar situaciones pueden ser configuradas mediante la interfaz web ofrecida por la plataforma. De esta manera se posibilita la involucración de usuarios no técnicos en el ciclo de vida del desarrollo en colaboración con el personal programador.

La metodología y la plataforma de desarrollo han sido validadas de manera exitosa con parejas de usuarios compuestas por un usuario técnico y un usuario no técnico. El grupo de usuarios técnicos se ha compuesto por programadores mientras que el grupo de los no técnicos se ha compuesto tanto por expertos en el dominio turístico como usuarios sin conocimientos de programación. Los resultados obtenidos sugieren que la involucración de expertos en el dominio en el proceso de desarrollo es posible y que dicha colaboración mejora el proceso de desarrollo y en consecuencia, el producto final. Finalmente, partiendo de los resultados obtenidos, se establecen implicaciones y líneas de trabajo futuras en el ámbito de los sistemas sensibles al contexto.

Así, la contribución principal de este trabajo es la nueva aproximación para el desarrollo de sistemas sensibles al contexto, basada en la colaboración entre expertos en el dominio y programadores. Esta contribución se basa tanto en las definiciones propuestas para los conceptos de *contexto* y *situación*, como en la metodología diseñada y la plataforma implementada para gestionar la información de contexto.

ABSTRACT

Information and services personalization is essential for an optimal user experience. This customization becomes even more important in mobile scenarios, where users have to interact with small devices and they usually require very specific information at a given time and place. This way, it is crucial for services to be able to acquire data about the user's context, process them in order to identify the user's situation and finally adapt the functionality of the system to that situation.

However, this personalization process is complex because many different tasks have to be taken into account. First, information from different heterogeneous and distributed data sources has to be acquired. Following, this information has to be transformed into a data model to make it machine-processable and it has to be managed in order to obtain the individual's situation and adapt the system to that situation. Consequently, diverse technologies are needed and a lot of technological challenges have to be faced by programmers in each development.

Over the last decade, several programming frameworks have been proposed in order to simplify the development of context-aware systems. However, there are still some gaps in the reviewed frameworks. On the one hand, not all of them are designed to support users' mobility. This is crucial since users' location is the primary context parameter in context-aware scenarios. On the other hand, the reviewed frameworks offer high-level application programming interfaces for skilled programmers. This makes the involvement of non-technical stakeholders in the development life-cycle, particularly domain experts, almost impossible. The participation of users that do not have programming skills but are experts in the application domain can speed up and improve the development process of context-aware systems.

In order to guide the development process, a methodology has been proposed which is called *Situation-Driven Development*. It has been designed in order to guide the identification and parameterization of the relevant situations that have to be detected by the context-aware system to be developed. The aim of this methodology is to promote the collaboration among programmers and domain experts in the development process.

Also, a development platform has been implemented, which is called *Context Cloud*. The aim of the platform is to make the development of context-aware systems easier, even for people that do not have technical skills. It supports the proposed methodology. This way, the platform provides a web front-end where all the features involved in the development of context-aware systems can be easily configured without coding any programming line. For instance, the context data model, the context information gathering process and the rules to identify users' situations can be configured using the web interface. This way, the involvement of domain experts in the development process is possible.

The methodology and the middleware platform have been successfully validated with pairs of technical users and non-technical users. The group of the technical users has been composed by programmers and the group of non-technical users has been composed by tourism domain experts and by general users without programming skills. The obtained results suggest that the involvement of domain experts in the development of context-aware systems is possible and that the collaboration among programmers and domain experts improves the development process. Finally, arising from these results, the thesis highlights some further implications for future works in the scope of context-awareness.

This way, the main contribution of this dissertation is the new proposed approach for the development of context-aware systems, based on the collaboration among domain experts and programmers. This main contribution is based on other contributions like the proposed definitions for *context* and *situation*, the designed development methodology and the implemented platform to manage context information.

LABURPENA

Informazio eta zerbitzu informatikoen pertsonalizazioa oso garrantzitsua da erabiltzaileei esperientzia on bat emateko. Pertsonalizazio hau askoz ere garrantzitsuagoa da mugikortasun inguruneetan, erabiltzaileek dispositibo txikiagoak erabili behar dituztelako eta normalki informazio konkretu bat bilatu behar dutelako, leku eta denbora konkretu batean. Hau dela eta, zerbitzuek erabiltzaileen testuinguru informazioa lortu, informazio mota hau prozesatu eta erabiltzailearen situazioa detektatu behar dute. Azkenik, sistemen portaera situazioaren arabera aldatu behar da.

Baina pertsonalizazio prozesua testuinguru informazioaren arabera egitea konplexua da. Alde batetik, testuinguru datuak lortu egin behar dira informazio iturburu ezberdin eta heterogeneoengatik. Informazio hau, transformatu egin behar da datu eredu baten arabera, sistemek kudeatu eta interpretatu ahal izateko. Bestetik, testuinguru datuen kondizio batzuen arabera, sistemek erabiltzailearen situazioa detektatu behar dute eta beraien portaera egokitu behar dute situazio horretara. Horregatik, programatzaileek teknologi, tresna eta ezagutza ezberdinak behar dituzte sistema hauen garapena egin ahal izateko.

Azken hamarkadan, programazio tresna asko proposatu egin dira testuinguru informazioa erabiltzen duten sistemen garapena errazteko, baina oraindik, zenbait gabetasun dituzte. Alde batetik, tresna hauek ez dira erabiltzaileen mugikortasuna kudeatzeko gai. Ezaugarri hau oso garrantzitsua da, kokapena testuinguru datu primarioa delako. Beste aldetik, programazio tresna hauek, programatzaileek bakarrik erabiltzeko gai diren interfaze ezberdinak eskaintzen dituzte. Hau dela eta, programatzaileak ez diren pertsonek garapen prozesuan parte hartzea oso zaila izaten da. Konkretuki, aplikazioaren domeinuan adituak diren pertsonek garapen prozesuan parte hartuz prozesu hau azkartu eta hobetuko litzateke. Erabiltzaile mota hauek sistemek detektatu behar dituzten situazioak hobeto identifikatzeko gai dira, baina normalki ez dute

programazio ezagutza. Ikerketa lan honen helburu nagusia, testuinguru informazioa erabiltzen duten sistemen garapena errazteko tresna bat garatzea da. Horrez gain, tresna honen bitartez, domeinu adituek garapen prozesuan parte hartzea da helburu.

Alde batetik, metodologia bat diseinatu egin da garapen prozesua gidatzeko, *Situation-Driven Development* edo Egoerarengatik Gidatutako Garapena izenekoa. Metodologia honek, sistemek detektatu behar dituzten egoeren identifikazio eta zehazpenean oinarritzen da. Metodologia honen helburua programatzaileen eta domeinu adituen arteko lankidetza bultzatzea da.

Horrez gain, garapen web ingurune bat inplementatu egin da, *Context Cloud* edo Testuinguru Hodeia izenekoa. Tresna honekin, testuinguru informazioa kudeatu daiteke programazio ezagutzarik gabe. Plataforma hau, diseinatutako metodologian oinarritu da. Web interfazearekin, testuinguru datu eredua, testuinguru informazioa eskuratzeko prozesua eta erabiltzailearen egoerak detektatzeko erregelak konfiguratu daitezke. Horrela, tresna honek domeinu adituen eta programatzaileen kolaborazioa errazten du.

Metodologia eta garapen web ingurunea balioztatu dira erabiltzaile bikoteekin, programatzaile eta ez programatzaileez osatuak. Programatzaileak ez direnak, turismo domeinuan adituak izan dira. Lortutako ondorioak adierazten dutenez, domeinu adituen partizipazioa posiblea da eta kolaborazio honek garapen prozesua hobetzen du. Azkenik, emaitza hauek direla eta, gerorako lan lerro batzuk proposatzen dira testuinguru informazioa erabiltzen duten sistemen arloan.

Horrela, lan honen ekarpen nagusia testuinguru informazioa erabiltzen duten sistemen garapenerako proposamen berria da, programatzaile eta domeinu adituen lankidetzan oinarrituta. Ekarpen honek, proposatutako testuinguru eta egoera definizioetan, metodologian eta sistemen garapena errazten duen web ingurunean oinarritzen da.

Chapter 1 Introduction

"Imagination is more important than knowledge.
Knowledge is limited. Imagination encircles the world"
Albert Einstein (1879 - 1955). German physicist.

Ubiquitous Computing is a new computing paradigm where all the everyday objects will have an embedded computer and they will be connected to the Internet, providing personalized services to users at any time and place (Weiser, 1991).

This vision is a reality in mobile environments. Our smartphones are connected to the Internet and they can provide us with information and services everywhere. But this mobile environment is very different from a desktop one. For instance, the screen is smaller, users are on the move and they usually require very specific information at a given time and place. This way, the customization of information is essential for an optimal user experience, especially in these kinds of mobile environments.

Nowadays, there are more than four billion of mobile devices all over the world and it is expected that by the year 2014, there will be more mobile devices connected to the Internet than desktop computers[1]. In addition to this, a report from Oracle (2011) states that the 69% of global mobile phone users have a smartphone (Android, iPhone, Blackberry or Windows Phone). These devices are replacing cameras, music players and Global Positioning System (GPS) tools. The 47% of mobile phone users have increased their data use over the year 2011. The 55% of them have reported having downloaded a free mobile application and the 25% have paid for an application. In this manner, the mobile device has become an open window to information and services at any time and

[1] http://www.ticbeat.com/wp-content/uploads/2011/04/mobile_infographic.jpg Last accessed 2012-19-03.

place. The challenge is to provide users with personalized information and services while on the move.

One of the key points in order to provide an optimal user experience in ubiquitous computing environments is the context in which the interaction between users and computers is carried out. This notion of context has been widely studied. One of the most referenced context definitions in the scientific community is the following.

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves" (Dey, 2001).

An example of context information is the location of the user. For instance, Location Based Services (LBS) (Seininger and Edwardes, 2006) use this context parameter in order to filter the results of a mobile search. In this manner, the information is automatically adapted based on the different context parameters on the move, providing the user with a better search experience.

Mobile customers are aware of the importance of the personalization of information in mobile environments. They are now more willing to share their location than they were a few years ago. In the year 2011, the 41% of the smartphone users have already elected to share their location with a mobile application (Oracle, 2011).

The industry is also aware of the significance of Context-Aware Computing as a scientific and technological research field. Gartner[2] has identified Context-Aware Computing as one of the top ten strategic technology trends for the year 2012 and they state that the advances in this area of research will create new user experiences.

Apart from that, new models of information and data sharing are becoming popular like the ones related to the Open Data movement (Bizer, 2009). These models' vision is to establish specific procedures in order to share data with everyone and promote the creation of innovative mashup services. In recent years, major web data sources such as Google or Yahoo have started to provide access to their databases through web application programming interfaces (APIs). The programmable web[3] initiative currently lists hundreds of such APIs in order to access very different context data types (e.g.

[2] http://www.gartner.com/technology/research/top-10-technology-trends/ Last accessed 2012-26-03.
[3] http://www.programmableweb.com/ Last accessed 2012-26-07.

sensor data, weather data). Cosm[4] and Nimbits[5] are companies that provide a web infrastructure in order to share data coming from different sensors. The democratization of data and the availability of services like the ones mentioned before are bringing new opportunities to create systems adapted to users' needs based on context data.

1.1. Problem statement

There are several challenges that have to be faced by the scientific community and the industry in order to provide the users with context-aware systems. This research work identifies two of these challenges.

- On the one hand, the development of context-aware systems is not a trivial task for programmers. These systems have to be able to obtain relevant context data in order to identify the situation of the user at a given time and place and adapt the behaviour of the system to that situation.
- On the other hand, it can be difficult for programmers to identify and parameterize the situations of the user that the system has to be able to detect, because they do not usually have the needed knowledge about the specific domain where the system has to be deployed.

As mentioned before, the implementation of context-aware systems is difficult for programmers because some technological challenges have to be considered and they are usually related to very different research fields (e.g. Human-Computer Interaction, Context-Aware Computing and Ambient Intelligence) and technologies (e.g. client-side and server-side technologies). Baldauf et al. (2007) identify some of the fundamental functionalities and requirements that have to be implemented in these developments.

- *Context data gathering*: the system needs to obtain context data about the user and her surroundings from distributed and heterogeneous context sources. These sources usually provide the programmer with very different ways to access context data. This information can be represented using different formats (e.g. eXtensible Markup Language or XML, plain text, JavaScript Object Notation or JSON).

[4] https://cosm.com/ Last accessed 2012-26-07.
[5] http://www.nimbits.com Last accessed 2012-26-07.

- *Context data processing*: the obtained context data has to be processed and harmonized in order to be aggregated by the system. Data transformation mechanisms are needed in order to perform the harmonization process.

- *Context data modelling*: a data model is needed in order to represent, store and manage contextual information more effectively. The data modelling language has to be expressive enough in order to represent data coming from heterogeneous sources and to make the harmonization process possible.

- *Context data management*: the data has to be managed by the system. This management involves the storage and updating of the received data based on the predefined data model. Also, inference mechanisms are needed in order to detect high-level context, that is, situations (e.g. cooking, walking, waiting for the bus) of the user.

- *System adaptation*: the system has to be able to react to the processed context data customizing the exposed services, offering a more satisfactory experience to the user.

In this manner, software toolkits that can ease the development of context-aware systems are needed. These toolkits have to provide the programmer with all the above functionalities in order to enable a rapid prototyping of the system to be implemented. Also, they have to be reusable in any new development, they have to be independent from the programming language to be used in the development of the context-aware system and they have to be flexible enough in order to be adapted to any application domain.

In addition to all the technological challenges that are involved in every new context-aware system development, there are some conceptual requirements that have to be specified as well. In this manner, it can also be difficult for programmers to identify the relevant situations that the system has to be able to detect. This task involves the parameterization of the needed context data to identify the required situations and the specification of the desired behaviours of the system once a situation is detected. All these parameters are dependent on the application domain (e.g. tourism, automotive industry, smart home) and usually programmers do not have the needed knowledge about all of them.

People that can overcome these drawbacks are domain experts, that is, people that are experts in a specific domain, but not necessarily experts in computer science, who use computer environments to perform their tasks (Costabile et. al, 2003). They can better identify the relevant situations for the system to be developed. For instance, let us imagine that a tourism office wants to create a mobile service in order to send information to visitors on the move based on their context (e.g. location, speed, weather, profile). People from the tourism office staff have the necessary knowledge about the tourism domain and they know exactly which situations of the visitor can be relevant for the mobile service and which information has to be sent to the visitor according to these situations. However, they do not necessarily have technical skills in order to be involved in the early stages of such developments. That is the reason why these toolkits need to be adapted to people with no programming skills. The involvement of domain experts in the development life-cycle of context-aware systems could optimize the development process and the final developed system could be more accurate to the initially specified requirements.

In addition to this, a collaborative development methodology is needed in order to guide the development process and involve domain experts in the development life-cycle of context-aware systems. The methodology has to be focused in the collaboration among domain experts and programmers and it has to be based on the definition of situations. This way, domain experts can collaborate with programmers in the detection of such situations using the functionalities offered by the software toolkit.

1.2. Work goals and hypothesis

The thesis claims that a new approach to the development of context-aware systems is possible providing new software toolkits and development methodologies in order to involve domain experts in the development life-cycle of such systems. There are four main research questions (RQ) that have been considered for this dissertation.

- *RQ1*. Can the context information be used in order to personalize systems and services in ubiquitous computing environments?
- *RQ2*. Can context data be managed in order to identify high-level situations using a software toolkit without programming?

- *RQ3*. Is it possible to involve domain experts in the development of context-aware systems?
- *RQ4*. Is it possible to promote the collaboration among domain experts and programmers using a methodology for the development and enhancement of context-aware systems?

The following hypothesis is proposed in order to provide an answer to these research questions.

It is possible to involve domain experts in the development life-cycle of context-aware systems in collaboration with programmers, by providing software toolkits and development methodologies adapted to people that do not have programming skills.

Two are the main goals (G) that have been set in order to validate the hypothesis.

- *G1*. To design a collaborative development methodology in order to involve domain experts in the development life-cycle of context-aware systems in cooperation with programmers.
- *G2*. To implement a context-aware software toolkit equally suitable for both programmers and domain experts.

The above mentioned goals also originate the following secondary goals.

- *G3*. To provide new definitions of *context* and *situation*.
- *G4*. To make the development of context-aware systems more universal and democratize the implementation of such systems.

1.3. Research methodology

The research presented in this thesis comprises theoretical and practical parts. In this manner, the dissertation has followed a research methodology in six stages (S).

- *S1*. Review of the existing approaches. The existing publications related to Context-Aware Computing have been revised. More precisely, existing context-aware toolkits and methodologies have been widely reviewed. Also, the notions of *context* and *situation* have been extensively studied. A critical evaluation has been carried out regarding the limitations, advantages and

disadvantages of the current approaches. This knowledge has also been reinforced attending specialized conferences[6].

- *S2*. Theoretical framework. Based on the results of the literature study of existing approaches, the requirements to achieve the previously mentioned goals have been established. These requirements are the basis for the definition of the conceptual framework.

- *S3*. Design. Based on the theoretical framework, a collaborative methodology to guide the development of context-aware systems has been designed. This methodology proposes the collaboration among programmers and domain experts as a key point to improve the development of such systems. Also, a toolkit to develop context-aware systems has been designed considering the drawbacks identified in the literature review, the defined theoretical framework and the development methodology.

- *S4*. Implementation. The software toolkit has been implemented as a middleware solution. It has been developed in order to be configured to detect situations of the user without coding any programming line.

- *S5*. Evaluation. The performance of the implemented middleware platform has been measured. The methodology and the toolkit have been evaluated with tourism domain experts and programmers. Also, the toolkit has been technically evaluated and compared with the existing approaches.

- *S6*. Dissemination of the results. The results obtained throughout the research process have been presented in different international conferences (a list of publications is available in Annex I). Also, other research centres have been visited[7] and meetings with other experts in the area of Context-Aware Computing have been held[8].

---

[6] International Conference on Pervasive Computing and Communications, PERCOM 2010 (http://www.percom.org/2010/ Last accessed 2012-26-03)

[7] The author was a visiting Ph.D. student at the Pervasive Media Studio (http://www.pmstudio.co.uk/) (Bristol) for three months (March - May 2012). The Studio is a multi-disciplinary lab exploring and producing pervasive media content, applications and services.

[8] The author presented this research work to people from Overlay Media, a company focused in the development of context-aware mobile services (http://www.overlaymedia.com/ Last accessed 2012-16-04)

1.4. Thesis outline

The organization of the remainder of this dissertation is divided as follows.

- Chapter 2 presents background in the field of context awareness. Previous *context* and *situation* definitions are analysed. Also, the classification and typologies of context data are discussed. Following, patterns of context data processing, software architectures and context data modelling techniques are presented. The different approaches in order to identify the situation of users are also described. Finally, the different development methodologies and context-aware toolkits are reviewed in order to identify challenges in the field and drawbacks or limitations in the existing approaches.

- Chapter 3 presents the conceptual and theoretical framework of this dissertation and its most important theoretic contributions. A new approach to the notion of context is presented. Also, a new definition for the notion of situation is explained. Following, the requirements of the development methodology are presented. Also, the requirements of the implemented toolkit are discussed. All these requirements are based on the proposed *context* and *situation* definitions and in the challenges identified in the state of the art.

- Chapter 4 provides the description of the designed methodology in order to promote the collaboration of domain experts and programmers in the development of context-aware systems. In this manner, the different stages of the methodology are described and an example of the usage of the methodology is explained.

- Chapter 5 presents the implemented toolkit in order to be used in the development of context-aware systems. The toolkit has been designed as a middleware platform, following the identified requirements and the detected drawbacks of the studied approaches. It has also been designed in order to make the involvement of non-technical users possible. First, the architecture of the toolkit is explained. Following, the functionalities of the toolkit are discussed and the usage with the designed methodology is explained.

- Chapter 6 provides the description of the evaluation of the methodology and the implemented software platform. The Chapter is further subdivided in two sub-chapters. Firstly, the evaluation of the performance of the platform in

terms of context data processing and reasoning capabilities is shown. Finally, the methodology and the results of the user evaluation are explained. A user evaluation has been carried out through a set of pairs composed by a programmer and a domain expert in order to see the acceptance of the methodology and the platform by real users. The empirical results obtained from the evaluations have been used to corroborate the hypothesis.

- Chapter 7 concludes with the major contributions of this research. It also shows how the objectives have been reached and the degree in which they have been fulfilled. Finally, some future research lines are outlined.

Capítulo 2 Estado del arte

"El que no sabe de nada, no duda de nada"
George Herbert (1593 - 1933). Poeta, orador y sacerdote inglés.

En este capítulo se recoge el estado del arte de aquellos ámbitos relacionados con los sistemas sensibles al contexto que tienen incidencia directa en el presente trabajo de investigación.

Por una parte, se describen aquellos aspectos relacionados con la información de contexto. Se recogen y analizan conceptos fundamentales como son los conceptos de *contexto*, *situación* y *sistema sensible al contexto*, revisando las distintas aproximaciones realizadas por diversos autores en las últimas décadas. Además, se realiza una descripción y análisis de la tipología y clasificación de la información de contexto. Se recogen también los patrones de computación utilizados para procesar y utilizar la información de contexto en beneficio de los sistemas cuyo comportamiento se quiere adaptar.

Se tratan también aspectos ligados a la infraestructura necesaria para dar soporte al desarrollo y ejecución de sistemas sensibles al contexto. De esta manera, se describe la arquitectura y las funcionalidades básicas que este tipo de sistemas deben ofrecer, revisando además, las diferentes estrategias utilizadas para modelar la información de contexto con el fin de que los sistemas computacionales puedan gestionarla y procesarla. Además, se analizan los diferentes entornos y librerías software para el desarrollo de sistemas sensibles al contexto cuyo objetivo es facilitar las labores de programación junto con las metodologías existentes enfocadas en la construcción de este tipo de sistemas.

Se concluye con un resumen de los temas tratados, resaltando las carencias encontradas en los diferentes puntos analizados en la revisión de la literatura.

2.1. Definición de conceptos fundamentales

En esta sección se realiza un repaso de las aproximaciones realizadas a lo largo de los últimos años a tres de los conceptos fundamentales en el ámbito de la computación sensible al contexto, como son *contexto*, *situación* y *sistema sensible al contexto*.

2.1.1. Contexto

Existen diferentes autores que han definido el concepto de contexto en sus trabajos de investigación. Todas estas definiciones pueden ser clasificadas en tres grandes grupos, en función de la información que contemplan o consideran los diferentes autores como parte del contexto.

Un primer grupo de definiciones toma como base el entorno en el que se produce la interacción entre el usuario y el sistema. Así, una de las primeras definiciones es la propuesta por Brown (1996) donde se recoge que contexto es *aquella información del entorno del usuario de la que el sistema tiene conocimiento*[9]. Una definición similar es la dada por Hull et al. (1997), donde se especifica que el contexto es la *información del entorno de la propia aplicación*. Lieberman y Selker (2000) tienen en cuenta que contexto es *todo aquello que afecta a la computación, excluyendo la información explícita de entrada y salida del sistema*[10]. Por su parte Chen y Kotz (2000) lo definen como *el conjunto de estados y configuraciones del entorno que determinan el comportamiento de una aplicación o que sirven para activar eventos de aplicación que son de interés para el usuario*[11]. Estas definiciones son bastante genéricas ya que no concretan lo que puede ser considerado como entorno y dejan demasiado ambiguo o abierto el concepto de contexto como para ser trasladado y modelado en un sistema computacional.

Otros autores toman como referencia la tarea o actividad que el usuario desempeña o pretende desempeñar. De esta manera, Henricksen (2003) define el contexto como *aquella información que es relevante para la consecución de una tarea concreta del usuario*[12]. Bucur et al. (2005) definen el contexto mediante una *finalidad u objetivo y una serie de atributos que son*

---

[9] *"Context can be a combination of elements of the environment that the user's computer knows about"* (Brown, 1996).
[10] *"Context can be considered to be everything that affects the computation except the explicit input and output"* (Lieberman y Selker, 2000).
[11] *"Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user"* (Chen y Kotz, 2000).
[12] *"The context of a task is the set of circumstances surrounding it that are potentially of relevance to its completion"* (Henricksen, 2003).

relevantes para llevar a cabo dicho objetivo[13]. En este segundo bloque de definiciones se considera la tarea o actividad a llevar a cabo por el usuario como elemento fundamental para la definición de contexto. Esta aproximación tiene como premisa que todo usuario tiene que tener alguna tarea o actividad definida, sobre la cual establecer qué tipo de información es relevante para la consecución de la misma. Este enfoque no encaja por lo tanto en aquellas interacciones en las que el usuario no tenga un objetivo concreto o no sea consciente del mismo de manera explícita, por lo que este tipo de aproximaciones se reducen a unos determinados escenarios que no pueden ser extensibles a otros dominios.

El tercer grupo de definiciones tienen en cuenta un mayor número de entidades participantes en el contexto. Por ejemplo, Schmidt et al. (1999) consideran el contexto como *el conocimiento sobre el usuario y los dispositivos, incluyendo su entorno, situación y localización*[14]. En la siguiente definición propuesta por Dey (2001) también se tienen en cuenta diferentes entidades relevantes participantes en el contexto. Así, se establece que contexto es *cualquier información que se pueda utilizar para caracterizar la situación de las entidades (personas, lugares u objetos) que se consideran relevantes a la hora de procesar la interacción entre el usuario y una aplicación, incluyendo en ese contexto tanto al mismo usuario como a la aplicación propiamente dicha*[15]. Uno de los puntos comunes de las últimas dos definiciones es la aparición del término *situación* en cada una de ellas. Mientras que la primera definición lo toma como parte del contexto, en la segunda aparece como una entidad de nivel superior respecto a la información de contexto. Así, este tercer bloque de definiciones tiene como pieza fundamental la situación de los usuarios, la cual es obtenida o detectada mediante la información de contexto conocida. En el apartado siguiente, se encuentran las diferentes definiciones revisadas en la literatura en relación a la situación de un usuario.

Todas estas definiciones muestran que el concepto de contexto no tiene un marco teórico común establecido. Esto es entendible, ya que para cada nueva aplicación, el contexto relevante será diferente y tendrá multitud de interpretaciones dependiendo de

---

13 *"Context can be considered as the factors that influence a certain decision"* (Bucur et al., 2005).

14 *"We use the term context in a more general way to describe the environment, situation, state, surroundings, task, location, situation, etc."* (Schmidt et al., 1999).

15 *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves"* (Dey, 2001).

los requerimientos concretos del sistema a implementar, el dominio en el que se sitúe y de la propia percepción de la información de contexto que tenga la persona que implemente el sistema.

2.1.2. Situación

En cuanto al término situación, existen también numerosos autores que han tratado de definirlo, habiendo definiciones muy generales y algunas más concretas.

Una de las primeras definiciones generales se refiere al *estado completo del universo en un instante de tiempo*[16] (McCarthy y Hayes, 1968). Existen diversos autores que han proporcionado definiciones más concretas para el concepto de situación. Brézillon (2002) trata una situación como un *subconjunto de la información de contexto general*[17]. Oppermann (2005) tiene una aproximación similar, considerando la situación como *aquellas características de contexto relevantes en un momento y lugar determinados*[18].

Por su parte, Coutaz y Rey (2002) utilizan una aproximación diferente, donde en vez de posicionar la situación en un nivel superior de abstracción sobre el contexto, determinan que una situación es un *conjunto de valores en un momento determinado y que diferentes situaciones a lo largo del tiempo son las que determinan el contexto de los usuarios*[19]. Así, al igual que en la definición de Schmidt et al. (1999), la situación forma parte del propio contexto.

Yau y Huang (2006) definen una situación como un *conjunto de contextos en un sistema a lo largo de un período de tiempo que afectan su comportamiento, considerando a su vez el contexto como cualquier propiedad relevante del entorno, del sistema o de los usuarios, que sea detectable e instantánea*[20]. Dey (2001) define la situación como *una descripción de los estados de diferentes entidades de contexto relevantes*[21].

Finalmente, cabe destacar que existen autores que utilizan el término *actividad* para referirse a la situación de los usuarios (Riboni y Bettini, 2010; Storf et al., 2009).

---

16 *"A situation is the complete state of the universe at an instant of time"* (McCarthy y Hayes, 1968).
17 *"A situation is a subpart of the overall context"* (Brézillon, 2002).
18 *"A situation is considered as the relevant context characteristics at a specific pointing time and space"* (Oppermann, 2005).
19 *"Our definition of context draws upon the notion of situation. A situation at time t is a state vector that is a set of observables at t. Context at t is a composition of multiple situations over a period of time"* (Coutaz y Rey, 2002).
20 *"A situation is a set of contexts in an application over a period of time that affects future system behaviour. A context is any instantaneous, detectable, and relevant property of the environment, the system, or users"* (Yau y Huang, 2006)
21 *"The situation abstraction is exactly that: a description of the states of relevant context entities"* (Dey, 2001).

Como puede observarse, aquí también existen diferentes aproximaciones al concepto de situación. Incluso se da el caso de utilizar diferente terminología para el mismo concepto, utilizando el término actividad para referirse a la situación. Un punto en común en las anteriores definiciones es la consideración de la situación en un nivel de abstracción superior al de la información de contexto, siendo a su vez dependiente de la misma.

2.1.3. Sistema sensible al contexto

Desde que aparecieron los primeros sistemas que utilizaban información de contexto, se han dado diferentes definiciones de los mismos. La primera vez que se habló de este tipo de sistemas fue de la mano de Schilit y Theimer (1994), donde los definían como *sistemas capaces de adaptarse a la localización del usuario, y las personas y objetos que le rodean, así como a los cambios que se producen en los mismos a lo largo del tiempo*[22]. Brown (1996) los define como *sistemas que tienen la capacidad de sentir y actuar en función de la información de su entorno*[23]. Dey (2000) considera que un sistema es sensible al contexto *si utiliza el contexto para proveer de información y/o servicios relevantes para el usuario, donde la relevancia viene determinada por la tarea del usuario*[24]. Finalmente, Baldauf (2006) los denomina como *sistemas que son capaces de adaptar su funcionalidad en función del contexto actual sin necesidad de intervención explícita por parte del usuario, con el objetivo de aumentar la usabilidad y efectividad*[25].

Hay autores que utilizan el término *sistemas sensibles a la situación* en vez de referirse al contexto. Además, se identifica un ámbito de conocimiento conocido como Conocimiento de la Situación o *Situation Awareness (SAW)* (Yau y Huang, 2006). De esta manera, un sistema sensible a la situación se define como un *sistema capaz de ser consciente de las situaciones y adaptarse a los cambios en las mismas*[26].

---

22 *"A context-aware system is a software that adapts its behaviour according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time"* (Schilit y Theimer, 1994).

23 *"Context-awareness is the ability of the computer to sense and act upon information about its environment"* (Brown, 1996).

24 *"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task"* (Dey, 2000).

25 *"Context-aware systems are able to adapt their operations to the current context without explicit user intervention and thus aim at increasing usability and effectiveness"* (Baldauf, 2006).

26 *"We consider situation awareness as the capability of being aware of situations and adapting the system's behaviour based on situation changes"* (Yau y Huang, 2006).

Parece claro entonces que, de manera general, se establece que un sistema sensible al contexto es aquel que es capaz de utilizar información de contexto para adaptarse a la situación de los usuarios.

2.2. Tipología de información de contexto

Existen autores que más que definir el concepto de contexto de manera explícita, lo definen mediante la especificación de distintos tipos de información de contexto que consideran relevante para sus trabajos. Uno de los primeros ejemplos que se pueden encontrar en la literatura relacionada es el de Abowd et al. (1996), donde toman la localización del usuario como único parámetro de contexto.

Otros autores como Chen (2004) tienen una aproximación particular al contexto basada en agentes software, teniendo en cuenta los deseos, creencias e intenciones de los mismos, los cuales representan a los usuarios. Por su parte, Gross y Specht (2000) introducen más tipos de información de contexto, considerando la localización e identidad de los usuarios, el tiempo (fecha y hora) y las características ambientales del entorno en el que se encuentran. Schilit et al. (1994) tienen en cuenta la localización del usuario, las personas y dispositivos del entorno y los cambios producidos en los mismos a lo largo del tiempo. Dey et al. (2001) en uno de sus trabajos también especifican una serie de tipos de información de contexto considerados relevantes, como la localización del usuario, la identidad, y el estado de las personas y objetos físicos y computacionales del entorno. Hess et al. (2003) también hacen una enumeración similar, teniendo en cuenta la localización del usuario, su identidad, el tiempo, el entorno o condiciones ambientales, la actividad del usuario y el dispositivo utilizado. Rahlff et al. (2001) introducen en este último listado el entorno social en el que está el usuario. Finalmente, Lee et al. (2007) tienen en consideración la localización del usuario, su identidad, el tiempo y la calidad del servicio.

Como se recoge en este repaso de la literatura, existen multitud de tipos de información de contexto que pueden ser utilizados para adaptar el comportamiento de los sistemas a las necesidades de los usuarios. En la Tabla 1 se muestra un resumen de cada uno de los tipos de contexto utilizados por los diferentes autores anteriormente mencionados. Además, se resaltan aquellos tipos de información de contexto que por haber sido utilizados en diversos trabajos de investigación se pueden considerar como relevantes en cualquier desarrollo de este tipo.

Tabla 1. Tipos de información de contexto

	Localización	**Identidad**	**Fecha y hora**	**Entorno ambiental**	**Entorno social**	**Dispositivo**	**Agentes software**	**Actividad**	**Calidad del servicio**
Abowd et al.	**x**								
Chen							x		
Gross et al.	**x**	**x**	**x**	**x**					
Schilit et al.	**x**				x	x			
Dey et al.	**x**	**x**		**x**					
Hess et al.	**x**	**x**	**x**	**x**		x		x	
Rahlff et al.	**x**	**x**	**x**	**x**	x	x		x	
Lee et al.	**x**	**x**	**x**						x

Los diferentes tipos de información de contexto utilizados dependerán por lo tanto de la finalidad del propio sistema a implementar, del dominio de aplicación en el que se quiera hacer uso del mismo y de los comportamientos que se quieran adaptar, por lo que resulta complejo realizar un listado exhaustivo de todos los tipos de información de contexto posibles.

Lo que sí resulta relevante, es que algunos tipos de información de contexto son más comúnmente utilizados, como son *la localización, la identidad de las entidades, el tiempo y las condiciones ambientales*, por lo que estos parámetros de contexto pueden ser considerados como primarios en cualquier tipo de sistema que quiera utilizar información de contexto para adaptar su comportamiento.

2.3. Clasificación de la información de contexto

Existen autores que han clasificado la información de contexto en diferentes categorías. La primera clasificación se da por parte de Schmidt et al. (1999) donde clasifican la información de contexto en factores físicos (localización, dispositivos y

condiciones del entorno), factores humanos (identidad del usuario, entorno social y actividad) y el tiempo (refiriéndose al histórico de contexto). Lieberman y Selker (2000) clasifican la información de contexto en información relativa al estado del usuario, su entorno físico, el entorno computacional y el histórico de contexto. Koripää et al. (2003) dispone de una clasificación basada en la localización, el tiempo, el entorno, el propio usuario y los dispositivos utilizados. Oh et al. (2006) utilizan los conceptos denominados 4W1H para clasificar la información de contexto en cinco categorías: quién *(Who)*, qué *(What)*, dónde *(Where)*, cuándo *(When)* y cómo *(How)*.

Por su parte, Schilit et al. (1994) disponen de dos clasificaciones. Una primera relativa a la información de contexto del usuario, que recoge información como su perfil, localización y el entorno social, y por otra parte el contexto físico, donde tienen cabida diferentes datos ambientales, como nivel de luz, ruido, etc. Chen y Kotz (2000) también clasifican la información en dos categorías. Una categoría de contexto activo, referida a aquella información de contexto que incide en el comportamiento de la aplicación, e información de contexto pasiva, donde la información de contexto es mostrada al usuario o almacenada para su posterior uso.

Gwizdka (2000) clasifica el contexto en externo e interno. El contexto externo engloba el estado del entorno del usuario. Por su parte, contexto interno es aquel que describe el estado del usuario. Henricksen e Indulska (2006) clasifican la información en estática, información capturada por los sensores, información proporcionada por el usuario e información inferida.

Finalmente existe una clasificación más general establecida por Guan et al. (2007), donde la información se divide en contexto de bajo nivel y contexto de alto nivel. Contexto de bajo nivel es toda aquella información obtenida sin necesidad de ningún tipo de procesamiento o combinación de datos. Por ejemplo, el nombre de una persona sería contexto de bajo nivel, de la misma manera que lo sería una lectura de temperatura procedente de un termómetro digital. Por su parte, contexto de alto nivel es un tipo de información abstracta resultante de procesar o razonar sobre el contexto de bajo nivel y obtener nueva información de contexto relevante. Un ejemplo sería el resultado de procesar una lectura de temperatura digital para, una vez cuantificada la señal, determinar si la temperatura en un entorno es baja, normal o alta. Esta clasificación se asemeja a la establecida por Ye et al. (2011), donde se establece como contexto primario los datos recogidos por los diferentes sensores y como contexto secundario la

información inferida a partir de los mismos. Esta división en contexto de bajo (primario) y alto nivel (secundario) encaja con la aproximación de situación antes revisada, estableciéndose de este modo la situación como una información de alto nivel obtenida a partir de información de contexto de bajo nivel.

De esta manera, y en base a las similitudes encontradas en las diversas clasificaciones revisadas, se puede establecer una clasificación de información de contexto de carácter general que se divide de la siguiente manera.

- *Usuario*: información relativa al usuario, como por ejemplo, la identidad del mismo, su localización, perfil, objetivos, entorno social (Buriano, 2006), etc.
- *Entorno físico*: toda información relativa al entorno del usuario, como por ejemplo, la temperatura, nivel de ruido, etc.
- *Dispositivos*: aquí se recogen los dispositivos con el que el usuario interactúa así como sus características.
- *Tiempo*: fecha y hora del día.

2.4. Patrones de computación sensible al contexto

Como se ha recogido anteriormente, los sistemas basados en el procesamiento del contexto obtienen y utilizan información de contexto para adaptar su comportamiento a dichos valores. Así, existen cuatro formas de utilizar la información de contexto en un entorno de computación (Gwizdka, 2000). Un sistema sensible al contexto puede adoptar uno o varios patrones, tal y como se indica a continuación.

- *El contexto como entrada adicional de datos*: este patrón considera el contexto como una entrada paralela a la información introducida por el usuario en su interacción. Un ejemplo puede ser una búsqueda de información aumentada automáticamente mediante información de contexto.

Fig. 1. Patrón de contexto como entrada adicional de datos (Gwizdka, 2000)

- *El contexto como modificador de entrada*: en este patrón, el contexto se utiliza para modificar la entrada de datos del usuario. Como ejemplo, se puede

considerar una interacción mediante mensajes de texto entre dos usuarios que hablan distinto idioma. El contexto en este caso sirve para modificar el mensaje del usuario emisor y traducirlo automáticamente al idioma del usuario receptor.

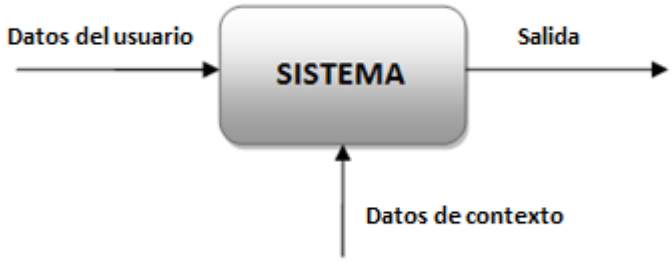


Fig. 2. Patrón de contexto como modificador de entrada (Gwizdka, 2000)

- *El contexto como información que vuelve al usuario*: representa un patrón de bucle cerrado y retroalimentado, en el cual, el contexto obtenido es presentado al usuario con el fin de poder utilizarlo para enriquecer la entrada del propio usuario. Por ejemplo, en un sistema de búsquedas, se podrían sugerir parámetros adicionales basados en información de contexto con el fin de que el usuario tuviera la opción de incluir dichos parámetros en la búsqueda.

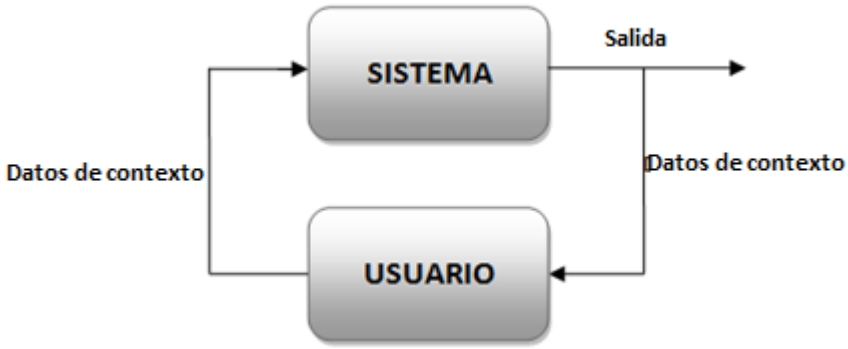


Fig. 3. Patrón de contexto como información de vuelta al usuario (Gwizdka, 2000)

- *El contexto como disparador de eventos*: este patrón deja de lado al usuario para centrarse en utilizar la información de contexto como principal entrada del sistema. Esa información, combinada con unas reglas de activación, permite al sistema determinar cuándo es necesario disparar un evento concreto en el sistema para adaptarlo al contexto de entrada. Un sistema puede utilizar este patrón para, por ejemplo, activar el envío de notificaciones adaptadas al contexto del usuario proporcionándole información adaptada a su situación.

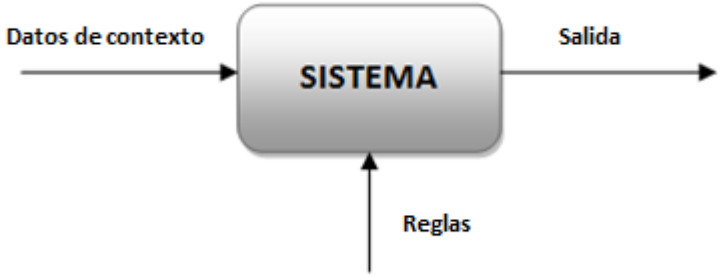


Fig. 4. Patrón de contexto como disparador de eventos (Gwizdka, 2000)

2.5. Arquitectura de los sistemas sensibles al contexto

Tal y como recogen Baldauf y Dustdar (2006) en un documento en el que repasan los aspectos clave de los sistemas sensibles al contexto, se pueden identificar una serie de parámetros de diseño y arquitecturales comunes a muchos de ellos. Estos conceptos proporcionan unas pautas a seguir a la hora de diseñar este tipo de sistemas.

De esta manera, se establece un denominador común a todos los sistemas sensibles al contexto en cuanto a arquitectura general se refiere. Tal y como se muestra en la siguiente figura, esta arquitectura se puede dividir en cinco capas diferenciadas e intercomunicadas: capa de sensores, capa de obtención de datos provenientes de los sensores, capa de procesamiento de datos, capa de gestión de datos y capa de aplicación.

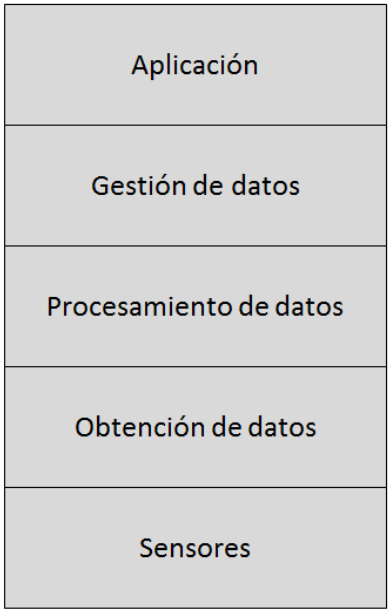


Fig. 5. Arquitectura de los sistemas sensibles al contexto (Baldauf y Dustdar, 2006)

A continuación se describen de manera funcional cada una de las capas que componen la arquitectura de los sistemas sensibles al contexto, desde el nivel inferior al nivel superior.

- *Sensores*: en esta capa se ubican los distintos tipos de sensores de los que el sistema obtiene la información de contexto necesaria. Estos sensores se pueden clasificar en tres tipos, tomando como elemento clasificatorio la manera en la que obtienen la información (Indulska y Sutton, 2003).
    - *Físicos*: estos sensores son de tipo hardware y son los más frecuentemente utilizados. Existen multitud de elementos sensoriales capaces de obtener información de muy diversa índole: micrófonos, acelerómetros, etc.
    - *Virtuales*: este tipo de sensores comprende aplicaciones o servicios web sobre los que se pueden obtener una serie de datos o información de contexto de interés. Por ejemplo, el correo del usuario, la agenda, un servicio web meteorológico, etc.
    - *Lógicos*: estos sensores son una combinación de los dos tipos anteriores, virtuales y físicos, junto con bases de datos adicionales para determinar información de contexto de más alto nivel.
- *Obtención de datos*: esta capa se encarga de recolectar los datos proporcionados por los sensores, bien mediante la API que pueda exponer el propio sensor o bien mediante otro tipo de interfaces con las que se pudiera interactuar (controlador, etc.). Esta capa es la intermediaria entre las fuentes de contexto (sensores) y niveles superiores de la arquitectura, por lo que proporciona una capa de abstracción a la hora de acceder a los datos.
- *Procesamiento de datos*: esta capa suele encontrarse en un servidor central al que llegan los datos obtenidos de la capa sensorial. Aquí se pueden realizar transformaciones de los datos para poder dar información de más alto nivel a las aplicaciones. Por ejemplo, unas coordenadas devueltas por un sistema GPS podrían ser traducidas de manera lógica al nombre o identificador de la habitación física donde se encuentra un usuario.

  También se podrían realizar composiciones de datos de diversas fuentes o sensores para dotar de mayor detalle a la información o corregir posibles conflictos existentes entre una misma información obtenida por sensores diferentes.

- *Gestión de datos*: en este nivel, los datos de capas inferiores son gestionados, y almacenados si procede, en base al modelo de datos elegido para la representación de la información de contexto. Es con esta capa con la que las aplicaciones y servicios tienen que interactuar, por lo que tiene que existir un API o interfaz para la comunicación entre ambas capas.

  Además esta capa es la que se encarga de establecer mecanismos de consulta y de inferencia sobre los datos de contexto obtenidos con el fin de componer situaciones abstractas de alto nivel más complejas.

- *Aplicación*: este nivel corresponde a los servicios, aplicaciones y sistemas que hacen uso de la información de contexto para adaptar su comportamiento a la misma.

Los módulos funcionales que debe contener la arquitectura de un sistema sensible al contexto están bien definidos y diferenciados. Cada uno posee sus características propias y trata de abstraer ciertas funcionalidades definidas a las capas superiores con el fin de concebir una división modular de la arquitectura. A nivel general, todos los sistemas analizados que se describen en la Sección 2.8 siguen esta distribución lógica en sus arquitecturas.

2.6. Modelado de la información de contexto

Como ya se ha comentado, para que la información de contexto pueda ser procesada por un sistema computacional se hace necesario definir un modelo de datos que sirva para representar dicha información y permita a las computadoras almacenar, manipular y procesar esta información. Existen cinco estructuras de datos comúnmente utilizadas, las cuales se describen a continuación (Strang y Linnhoff-Popien, 2004).

- *Pares "clave-valor"*: el modelado de la información contextual mediante pares es prácticamente el modelado más simple que se puede realizar de la misma. El método de modelado se basa en una serie de pares que representan el contexto de una entidad, con el fin de que un algoritmo o proceso pueda analizar la información contextual en busca de unos valores determinados para unas claves concretas. Si bien se trata de un modelado funcional, su excesiva sencillez presenta importantes limitaciones a la hora de gestionar la información contextual cuando se trata de sistemas complejos. Esta técnica

de modelado se ha utilizado por ejemplo en el trabajo realizado por Schilit et al. (1994), donde se modela la localización de una entidad mediante pares "clave-valor" del tipo: "localización = oficina".

- *Modelos basados en etiquetas*: el modelado de la información contextual puede también realizarse mediante lenguajes basados en etiquetas, tales como XML. El modelado de contexto usando dichos lenguajes tiene ventajas a la hora de gestionar la información, dado que introduce la noción de jerarquía dentro de la información contextual, aunque el procesado de la información etiquetada pueda ser menos eficiente que el procesamiento de contextos basados en pares. ConteXtML (Ryan, 1999) es un claro ejemplo de modelado contextual basado en etiquetas.

- *Modelo orientado a objetos*: la técnica de modelado del contexto mediante orientación a objetos surge de manera natural al realizar desarrollos con lenguajes de programación orientados a objetos. Además, modelar el contexto usando técnicas de orientación a objetos ofrece ventajas como la encapsulación, reusabilidad, herencia, etc. Un ejemplo de dicha aproximación es el proyecto Hydrogen (Hofer et al., 2002). La principal ventaja de este tipo de modelado radica en la facilidad de uso para el programador, mientras que su principal desventaja viene de la dificultad de compartir la información contextual entre fuentes, limitando también el procesado de la misma.

- *Modelos basados en reglas lógicas*: estos modelos se basan en reglas, expresiones y hechos que se utilizan para definir un contexto determinado, donde se añaden nuevas reglas y hechos para poder modelar nuevos tipos de contexto. Se pueden utilizar mecanismos de inferencia o razonamiento para procesar y obtener nuevos hechos sobre los ya recopilados, mejorando así la comprensión del entorno por parte del sistema. Por ejemplo, el sistema implementado por Bacon et al. (1997) hace uso de estas técnicas, expresando la información contextual de localización mediante hechos que se almacenan en un sistema basado en reglas lógicas.

- *Ontologías*: las ontologías son formalizaciones de información que representan conceptos y las relaciones entre ellos, proporcionando un alto

nivel de expresividad (Gruber, 1993). Además, mediante el uso de motores de inferencia, razonadores lógicos o motores de reglas, permiten aumentar el conocimiento que el sistema de computación maneja. Un ejemplo de modelado contextual basado en ontologías es la propuesta CONON (Wang et al., 2004), donde se desarrolla un modelo de tipo ontológico por las ventajas que dichos modelos ofrecen a la hora de compartir el conocimiento, inferir nuevo conocimiento lógico y reutilizar la base de conocimiento.

Algunos autores afirman que las ontologías son la mejor aproximación debido a la expresividad que ofrecen (Strang y Linnhoff-Popien, 2004; Chen et al., 2003). No obstante, cabe destacar que aunque los autores recomiendan el uso de ontologías, reconocen que todas las técnicas son igualmente válidas. La elección del modelo de representación vendrá dada por el nivel de expresividad que el sistema a implementar requiera, ya que dependiendo de una mayor o menor expresividad, pueden ser aplicables unos u otros sistemas de representación.

Una de las características que el modelo de datos debe cumplir es la posibilidad de ser extendido de manera dinámica (Henricksen, 2003). Este factor se debe a la cantidad de datos dinámicos que este tipo de sistemas manejan, lo que hace que el modelo de datos sea variable en función de las nuevas fuentes de contexto que puedan ser introducidas en el sistema. Una rápida integración de dichas fuentes y adecuación del modelo de datos a las mismas es vital para disponer de sistemas tolerantes a los cambios.

2.7. Mecanismos para la detección de situaciones

Además de disponer de un modelo de datos con el que representar la información de contexto y que pueda ser procesada por sistemas computacionales, es necesario el establecimiento de mecanismos que puedan detectar, partiendo de los datos recogidos en el modelo, las diferentes situaciones que sean relevantes para el sistema a implementar.

En la literatura revisada se han encontrado dos tipos de mecanismos utilizados (Bettini et al., 2010; Yu et al., 2011), cada uno con diversas técnicas, como son los basados en la especificación de las situaciones y los basados en metodologías de aprendizaje.

Los mecanismos basados en la especificación de situaciones funcionan en base a la configuración de reglas lógicas de primer orden con las que sistemas de razonamiento pueden inferir conocimiento detectando así las situaciones definidas mediante dichas reglas (Chen et al., 2003; Beer et al., 2007). Algunos de estos sistemas se fundamentan en un modelo basado en ontologías (Bikakis et al., 2009; Wang et al., 2004).

En cuanto a los mecanismos basados en el aprendizaje, existen un gran número de técnicas, siendo las más utilizadas las basadas en clasificadores Naïve Bayes (Patterson et al., 2003), Redes Bayesianas (Ranganathan et al., 2004) o modelos ocultos de Markov (Kasteren et al., 2008). Estos mecanismos tienen el inconveniente de que se necesita una gran cantidad de datos para configurar un modelo fiable con el que identificar las situaciones requeridas. Además, los modelos resultantes son difícilmente adaptables en tiempo real a cambios que se puedan dar en los requisitos de información de contexto, como la adición de nuevas fuentes de contexto que proporcionen nuevos datos a ser procesados.

2.8. Entornos de desarrollo para la programación de sistemas sensibles al contexto

Uno de los primeros servicios que utilizó información de contexto fue un sistema de localización ideado en un entorno de oficinas denominado *Active Badge Location System* (Want et al., 1992). El citado sistema permitía hacer uso de la información referente a la localización de un usuario para determinar la manera más adecuada de comunicarse con él. Ya en los años 90, se desarrollaron guías turísticas que utilizaban la localización del usuario como parámetro de contexto con el fin de proporcionar información adaptada a dicha localización (Abowd et al., 1997; Sumi et al., 1998).

A medida que este tipo de sistemas se iban dando a conocer, su complejidad iba en aumento ya que se tomaban en cuenta más tipos de información de contexto. Así, empezaron a aparecer los primeros entornos de desarrollo orientados a ofrecer todas las funcionalidades e infraestructura necesarias para que los programadores pudieran implementar este tipo de sistemas de una manera más sencilla y extensible.

En los siguientes apartados se describen los entornos de desarrollo más significativos revisados en la literatura y posteriormente se realiza un análisis comparativo de los mismos.

2.8.1. Entornos y librerías de desarrollo

A continuación se realiza una descripción de los diferentes entornos de desarrollo analizados.

2.8.1.1. ***Context Toolkit***

Este entorno de programación es una de las primeras aproximaciones para el desarrollo de sistemas sensibles al contexto implementada en lenguaje Java (Dey et al., 2001). En la Figura 6, se pueden diferenciar los primeros módulos funcionales específicos de este tipo de sistemas contextuales, divididos en diferentes capas que conforman una de las primeras arquitecturas de los sistemas sensibles al contexto.

En un nivel inferior se encuentran los sensores, que se encargan de obtener diferentes tipos de información de contexto. Posteriormente los proveedores o *widget* son los que acceden a los sensores con el fin de obtener la información y exponerla a niveles superiores de la arquitectura. Esta información está modelada en base a pares "clave-valor" y codificada en formato XML. Los módulos de gestión aglutinan la información y el intérprete posibilita la realización de inferencias simples. En el último nivel se encuentra la capa de aplicaciones que utiliza la información de contexto procesada en el sistema.

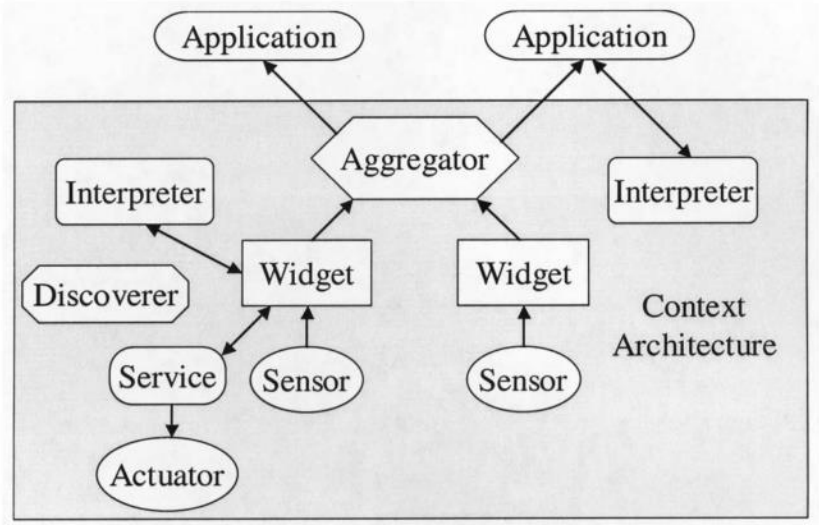


Fig. 6. Arquitectura del entorno de desarrollo *Context Toolkit* (Dey et al., 2001)

2.8.1.2. **SOCAM**

El entorno SOCAM *(Service-oriented Middleware for building Context-Aware Services)* (Gu et al., 2005) es una infraestructura basada en servicios que permite desarrollar

aplicaciones sensibles al contexto mediante el API expuesto. Este API proporciona servicios para consultar o suscribirse a una determinada información de contexto. El modelo de datos de contexto está estructurado en base a ontologías. La infraestructura SOCAM clasifica la información de contexto entre información directa e indirecta. Dentro de la información directa se encuentra la información obtenida por los sensores que se registran en la infraestructura. También se encuentra en este nivel la información definida por el usuario (preferencias de usuario). Se considera indirecta, la información que es inferida o deducida partiendo de otra información contextual mediante sistemas de inferencia aplicados a los modelos semánticos.

La arquitectura se divide en diferentes capas tal y como se muestra en la Figura 7. En la capa inferior se encuentran los proveedores de contexto *(Context Providers)* los cuales se encargan de obtener la información de los sensores, en una capa intermedia se encuentra el intérprete de contexto *(Context Interpreter)*, que incluye una base de conocimiento *(Context KB)* y un sistema de razonamiento *(Context Reasoner)* para realizar procesos de inferencia. En el nivel superior se encuentran todos aquellos servicios que utilizan la infraestructura.

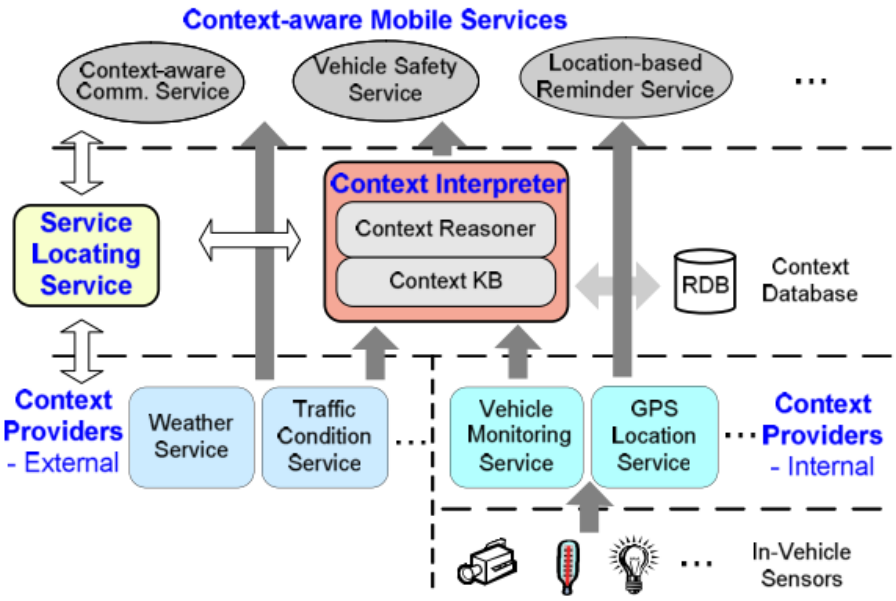


Fig. 7. Arquitectura del entorno de desarrollo SOCAM (Gu et al., 2005)

2.8.1.3. **CoBra**

El sistema CoBra *(Context Broker Architecture)* (Chen et al., 2003) utiliza una arquitectura basada en agentes software distribuidos con el fin de que sean estos los que

accedan a la información de contexto dispersa en diversos sensores. Utiliza una ontología como modelo de datos del sistema. Esta aproximación basada en ontologías posibilita que los agentes intercambien conocimiento contextual y puedan razonar sobre el mismo. Este proceso de inferencia se da gracias al motor de razonamiento basado en reglas que contempla la arquitectura. El elemento central de la arquitectura que se propone es el bróker de contexto o *Context Broker*, el cual orquesta todos los servicios ofrecidos por la infraestructura para que los diferentes agentes puedan operar.

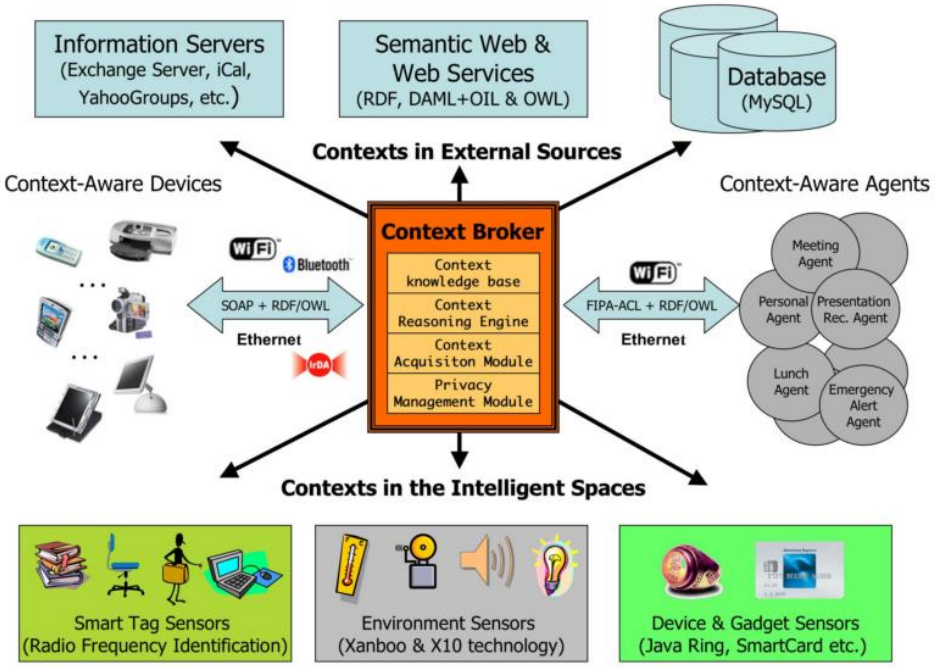


Fig. 8. Arquitectura del entorno de desarrollo CoBra (Chen et al., 2003)

2.8.1.4. **JCAF**

JCAF *(Java Context Awareness Framework)* (Bardam et al., 2005) es un entorno de programación para lenguaje Java que proporciona un API de alto nivel y un entorno de ejecución para las aplicaciones desarrolladas. La estructura de datos utilizada para representar la información de contexto es un modelo de datos orientado a objetos. El funcionamiento se basa en detectores de eventos de contexto *(Entity Listener)* que actúan en función de los datos de contexto recibidos *(Context Monitor)* para desencadenar diferentes acciones *(Context Actuator)*. El sistema no contiene ningún mecanismo de razonamiento ni inferencia para poder obtener información de contexto de alto nivel. Su arquitectura se divide en dos módulos. Por una parte, un módulo donde se encuentran los componentes que obtienen los datos de los sensores *(Context Client Tier)* y por otra parte un entorno de ejecución para los servicios desarrollados *(Context Service Tier)*.

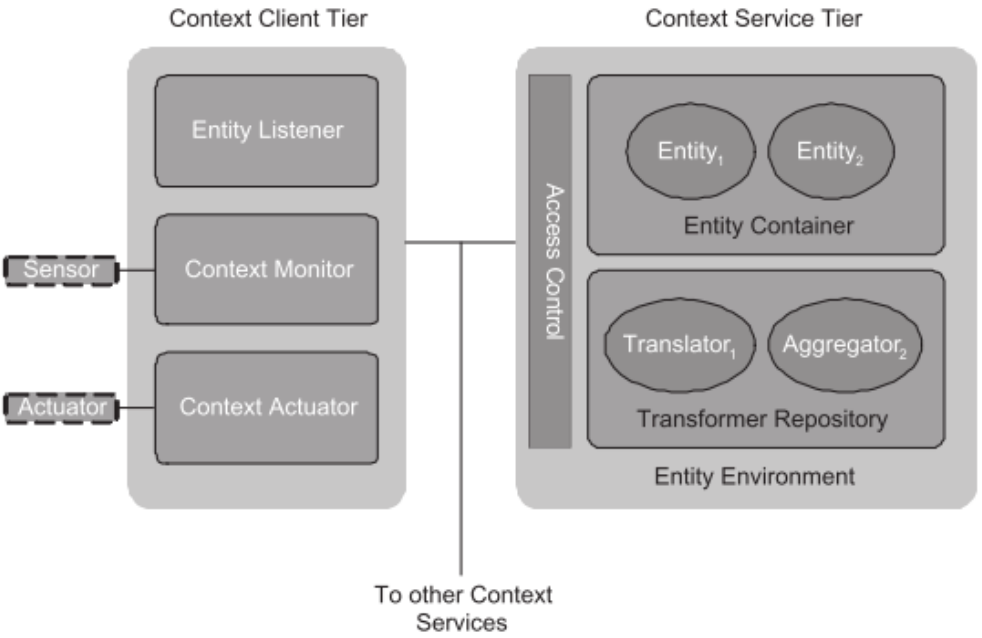


Fig. 9. Arquitectura del entorno de desarrollo JCAF (Bardam et al., 2005)

2.8.1.5. ***Semantic Space Toolkit***

Esta infraestructura permite desarrollar sistemas sensibles al contexto, utilizando un modelo de datos basado en ontologías (Wang et al., 2004). Para inferir contexto de alto nivel se utilizan una serie de reglas lógicas sobre la base de conocimiento *(Context Knowledge Base)*, que dan como resultado contextos de alto nivel que las aplicaciones pueden utilizar para adaptar su comportamiento. Estos contextos de alto nivel tienen que ser consultados por las aplicaciones cada vez que quieran obtener información para adaptar su comportamiento. Además, la infraestructura permite la suscripción por parte de las aplicaciones finales a los diferentes cambios que se produzcan en la información de contexto.

La arquitectura que se muestra en la Figura 10, sigue la división en capas típica de los sistemas sensibles al contexto, donde se puede encontrar la capa de sensores, la capa de acceso a los sensores *(Wrapper)*, el aglutinador de información de contexto *(Context Aggregator)* junto con los mecanismos de razonamiento *(Context Reasoner)* y consulta de información de contexto *(Context Query Engine)* sobre la base de conocimiento.

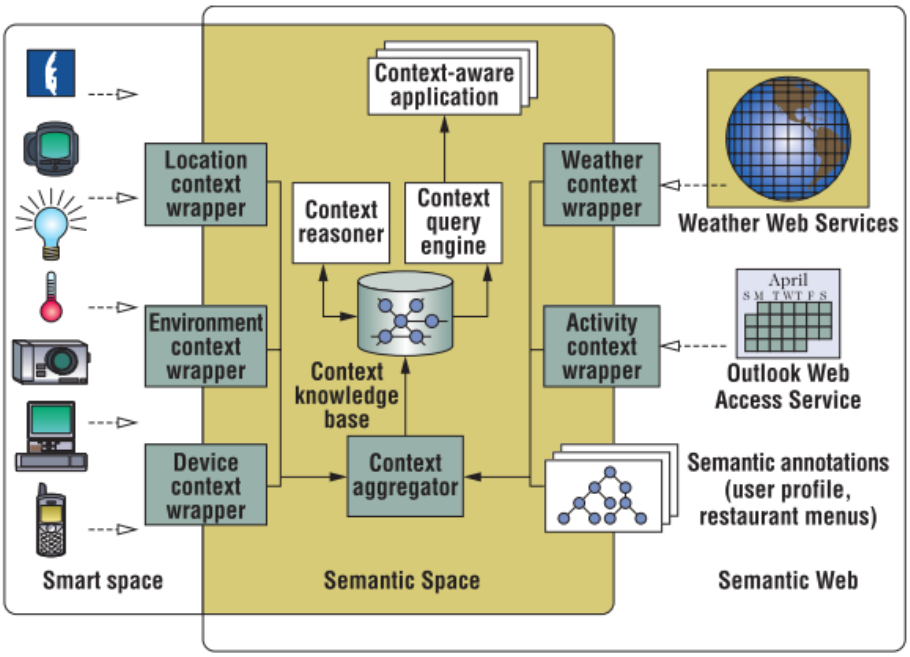


Fig. 10. Arquitectura del entorno de desarrollo *Semantic Space Toolkit* (Wang et al., 2004)

2.8.1.6. **CASS**

CASS *(Context-Awareness sub-structure)* (Fahy y Clarke, 2004) es una infraestructura que utiliza también detectores de eventos *(Sensor Listener)* que capturan cambios en la información de contexto proveniente de los sensores *(Sensor)*. La información se representa mediante bases de datos relacionales *(Database)*, utilizando mecanismos de interpretación *(Interpreter)* e inferencia basados en reglas para reconocer situaciones de alto nivel *(Rule Engine)*.

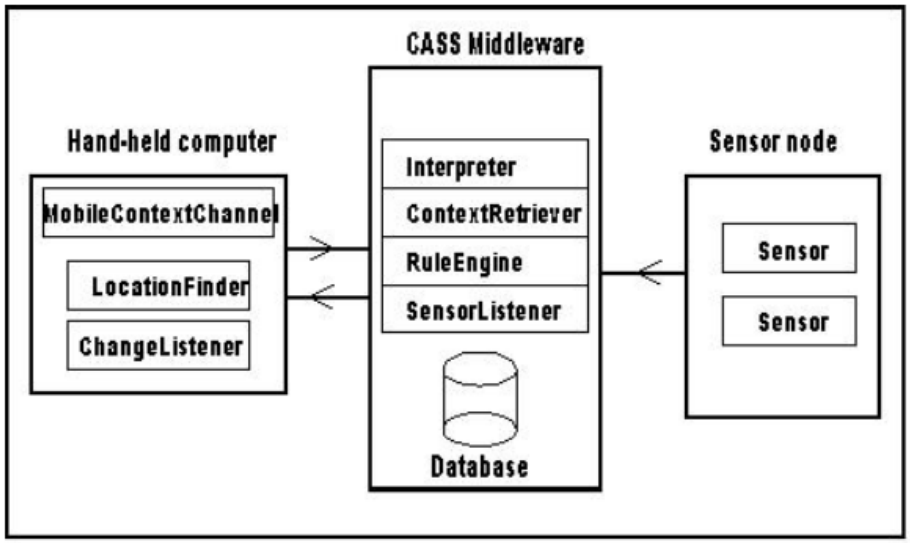


Fig. 11. Arquitectura del entorno de desarrollo CASS (Fahy y Clarke, 2004)

2.8.1.7. **CMF**

CMF *(Context Management Framework)* (Korpipää et al., 2003) es una librería de programación orientada a dispositivos móviles, por lo tanto las fuentes de contexto se localizan en el propio dispositivo. Utiliza un modelo de datos en lenguaje RDF y clasificadores Bayesianos para reconocer situaciones de alto nivel. Estas situaciones de alto nivel son relativamente pobres, ya que el sistema no permite añadir ninguna fuente externa de contexto limitándose únicamente a las proporcionadas por el dispositivo móvil.

La arquitectura que se muestra en la siguiente figura muestra los componentes de este entorno de programación, donde los módulos principales son los correspondientes a la obtención *(Change detection service)* y gestión de la información de contexto *(Context Manager)*.

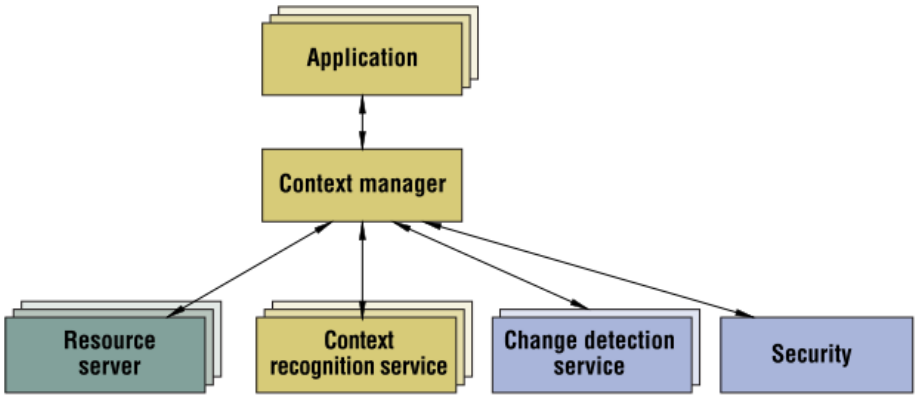


Fig. 12. Arquitectura del entorno de desarrollo *Context Management Framework* (Korpipää et al., 2003)

2.8.1.8. ***Hydrogen***

Este entorno (Hofer et al., 2002) está pensado también para ser ejecutado en entornos móviles basados en J2ME *(Java to Micro Edition*[27]*)*. Se divide en cuatro capas, donde se encuentran la capa de sensores, la capa de obtención de datos *(Adaptor Layer)*, la capa de gestión de datos *(Management Layer)* y la capa de aplicación *(Application Layer)*.

El modelo de datos utilizado es un modelo de datos orientado a objetos basado en lenguaje Java. No soporta ningún tipo de inferencia y las únicas fuentes de contexto se

[27] http://www.oracle.com/technetwork/java/javame/index.html Último acceso 26-07-2012

limitan a las disponibles en el propio dispositivo móvil, como por ejemplo la hora o la localización.

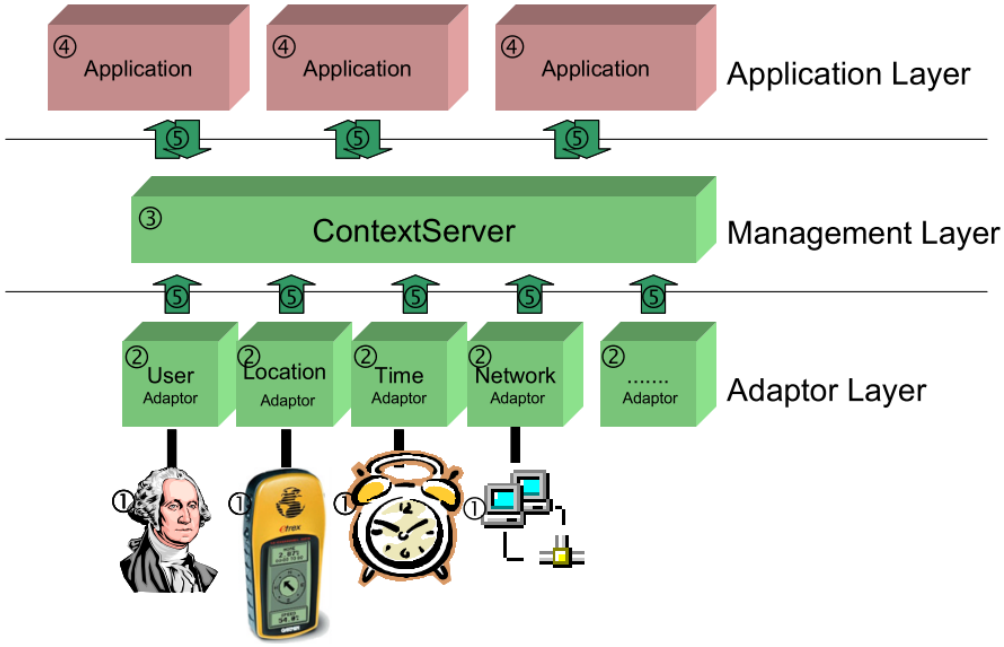


Fig. 13. Arquitectura del entorno de desarrollo *Hydrogen* (Hofer et al., 2002)

2.8.1.9. **DiaSuite**

Esta herramienta de desarrollo se compone de diferentes elementos (Cassou et al., 2010). Por una parte, dispone de un lenguaje de programación específico para la implementación de los componentes y comportamientos de los servicios contextuales, así como un compilador para dicho lenguaje. Este lenguaje permite también definir entidades de contexto mediante taxonomías. Por lo tanto, el modelado se realiza mediante orientación a objetos, teniendo un sistema de inferencia basado en reglas lógicas.

Además, proporciona un entorno visual para simular escenarios en dos dimensiones y condiciones de contexto dinámicas (cambios de temperatura, cambios en la red de celdas de telefonía móvil, etc.) con el fin de ayudar en la validación de los servicios. La filosofía de este entorno persigue un desarrollo colaborativo, donde entran en juego diferentes actores de diferentes perfiles. Desde expertos del dominio, validadores, desarrolladores, administradores de sistemas y analistas funcionales.

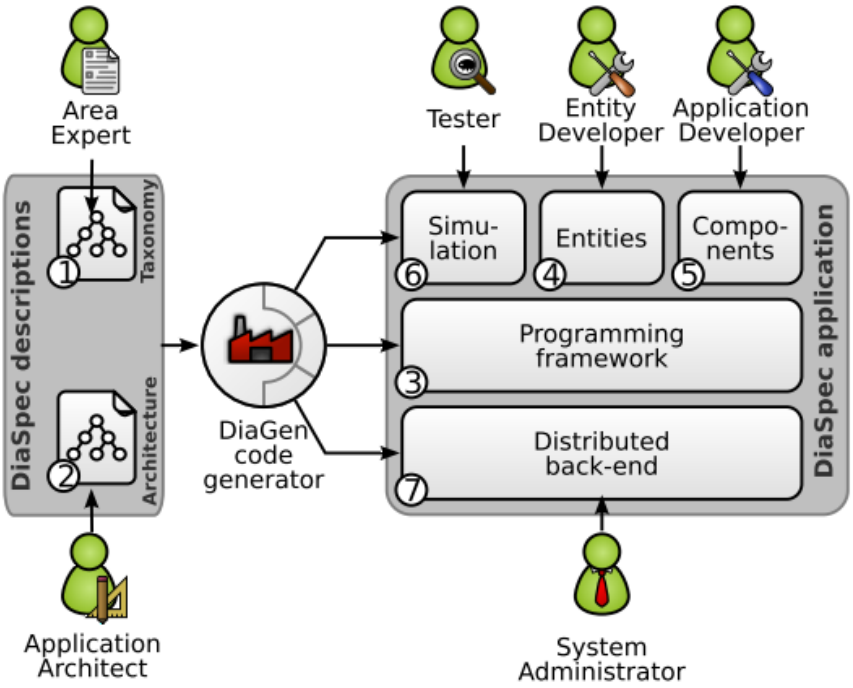


Fig. 14. Arquitectura del entorno de desarrollo *DiaSuite* (Cassou et al., 2010)

2.8.1.10. **OPEN**

El entorno OPEN (Guo et al., 2010) está pensado para que tanto usuarios expertos en programación, como los no expertos puedan desarrollar prototipos que utilicen información de contexto de manera colaborativa.

Se representa la información de contexto en base a un modelo basado en ontologías que es definido por expertos, sobre el que se definen reglas lógicas para identificar situaciones. Este modelo basado en ontologías se almacena y gestiona en el gestor de contexto *(Context Manager)*, donde se agregan los diferentes datos de contexto obtenidos por los proveedores *(Context Provider)* y los diferentes adaptadores *(Wrapper)* programados para cada tipo de fuente de contexto de la que se quiera obtener información.

Las reglas definidas pueden desencadenar diferentes acciones sobre elementos fijos que proporciona la infraestructura, como por ejemplo aplicaciones multimedia. De esta manera las acciones se limitan a reproducción de audio, video, etc.

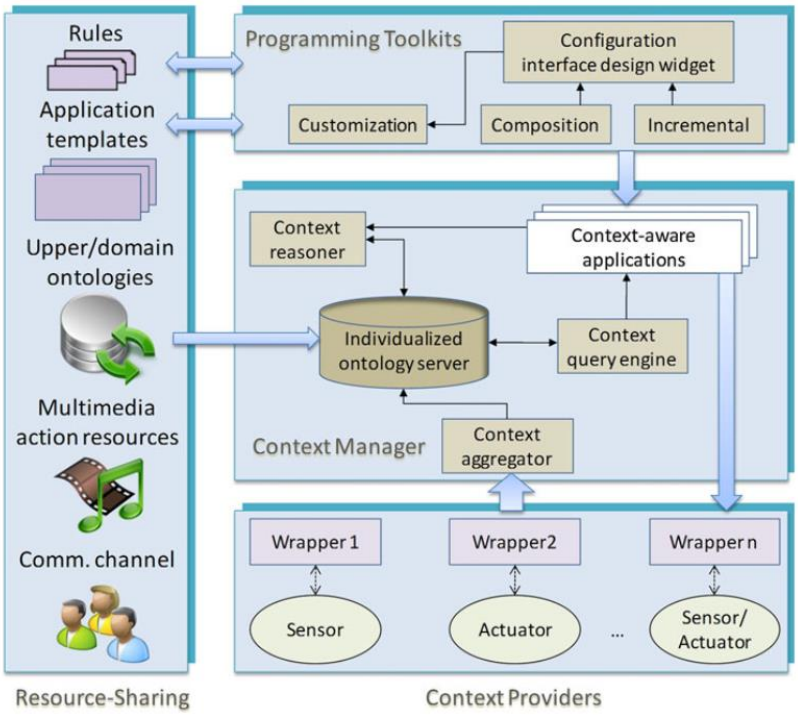


Fig. 15 Arquitectura del entorno de desarrollo OPEN (Guo et al., 2010)

2.8.2. Comparativa y análisis de los entornos de desarrollo

En esta sección se realiza una comparativa de todos los entornos de desarrollo revisados anteriormente, en base a una serie de características fundamentales que se definen a continuación.

- *Modelo de datos*: especifica el modelo de contexto que se utiliza para representar la información de contexto.

- *Extensible*: un aspecto importante en un entorno tan cambiante como el de los sistemas que utilizan información de contexto, es el de posibilitar la extensión en tiempo real del modelo de datos en función de los valores que interesen en cada momento (Henricksen, 2003). Además, los comportamientos del sistema sensible al contexto definidos tienen que poder adaptarse también dinámicamente. Así, este parámetro indica si el modelo de contexto y los comportamientos definidos son extensibles de manera dinámica.

- *Gestión automática de la información de contexto*: este parámetro indica si la gestión de la información de contexto se realiza de manera automática o si de lo contrario, es el propio programador el que tiene que preocuparse de la

gestión de los datos. En entornos tan dinámicos como los relativos a los sistemas sensibles al contexto, es interesante que el propio entorno de desarrollo sea capaz de gestionar los cambios en la información de contexto de manera autónoma. De esta manera, todas las actualizaciones del modelo de datos, tienen que poder realizarse automáticamente. Además, el entorno tiene que poder controlar el perfecto funcionamiento de las fuentes de contexto y eliminar la información de aquellas fuentes que ya no estén operativas.

- *Mecanismos de razonamiento*: indica si se disponen de mecanismos de razonamiento para poder inferir conocimiento de alto nivel partiendo de los datos de contexto recogidos. Este tipo de mecanismos posibilitan la transformación de información contexto de bajo nivel a contexto de alto nivel.

- *Soporte para movilidad*: indica si el entorno ofrece algún tipo de funcionalidad para gestionar la movilidad de las entidades de contexto. En escenarios en movilidad es relevante que el entorno pueda gestionar la localización de las entidades relevantes a la hora de detectar la situación de las mismas.

- *Usuarios no técnicos*: indica si es un entorno apto para poder ser utilizado por usuarios sin conocimientos de programación.

- *Entorno visual*: indica si la herramienta proporciona, además de un API de alto nivel orientado a programadores, un entorno visual de desarrollo en el que se minimicen las interacciones programáticas con el entorno de desarrollo.

- *Solución web*: indica si el entorno es accesible y configurable desde cualquier navegador web con acceso a Internet.

Tabla 2. Comparativa de las diferentes librerías y entornos de programación analizados

	Modelo	**Extensible**	**Gestión automática**	**Razonamiento**	**Movilidad**	**Usuario no técnico**	**Entorno Visual**	**Web**
Context Toolkit	Pares "clave-valor"	No	No	Sí	No	No	No	No
SOCAM	Ontologías	No	No	Sí	No	No	No	No
CoBra	Ontologías	No	No	Sí	No	No	No	No
JCAF	Orientado a Objetos	No	No	No	No	No	No	No
Semantic Space Toolkit	Ontologías	No	No	Sí	No	No	No	No
CASS	Relacional	No	No	Sí	No	No	No	No
CMF	Ontologías	No	No	Sí	Sí	No	No	No
Hydrogen	Orientado a Objetos	No	No	No	Sí	No	No	No
DiaSuite	Orientado a Objetos	No	No	Sí	No	Sí	Sí	No
OPEN	Ontologías	No	No	Sí	No	Sí	Sí	Sí

Tal y como refleja la Tabla 2, existen todavía carencias en este tipo de herramientas de cara a la implementación de sistemas sensibles al contexto[28]. En cuanto a la gestión de la información de contexto, ninguno de los entornos o librerías analizados disponen de una gestión automática de la información de contexto, teniendo que obligar al programador a realizar esta gestión. Con lo cual, se requiere más trabajo de implementación y una mayor exposición a posibles errores. Tampoco posibilitan la extensión o modificación del modelo de datos en tiempo de ejecución ni de los comportamientos definidos, por lo que los sistemas implementados no pueden adaptarse en tiempo real a modificaciones y nuevos requerimientos en la información de contexto que manejan.

Además, no todas las herramientas están pensadas para soportar la movilidad del usuario. Solamente *Hydrogen* y *Context Management Framework* están orientadas a un entorno móvil. Sin embargo, esta orientación solamente se basa en que son librerías software que se despliegan en el propio dispositivo móvil. Este aspecto limita la funcionalidad y potencia de la librería ya que solamente se tiene acceso a la información del dispositivo móvil y la capacidad de procesamiento se limita a la ofrecida por el propio dispositivo. Además, ningún entorno ofrece la posibilidad de realizar operaciones en base a la localización de las entidades, ya que carecen de un módulo geoespacial que lo posibilite. Esta es una carencia importante ya que, tal y como se recoge en secciones anteriores, uno de los datos de contexto más importantes es la localización de las entidades de contexto, como por ejemplo, los propios usuarios del sistema final.

Por su parte, todos los entornos de programación, a excepción de DiaSuite y OPEN, están orientados al programador y no ofrecen entornos visuales de programación, limitándose a ofrecer APIs de programación de alto nivel, lo que imposibilita la participación de personas no técnicas en el ciclo de vida del desarrollo de la solución. Involucrar a este perfil de personas en el proceso de definición de situaciones del usuario, en base a los datos de contexto obtenidos, puede acelerar y mejorar el proceso de desarrollo de servicios contextuales.

[28] En la Sección 5.9.2 se realiza un estudio comparativo de estas alternativas con la propuesta en este trabajo de investigación, la cual da solución a las carencias encontradas en la revisión del estado del arte.

Aun así, DiaSuite tampoco ofrece una aproximación en la que usuarios no programadores puedan participar en el desarrollo de la solución, ya que el grueso de la programación se realiza mediante un lenguaje específico, parecido a Java, en el que se definen los diferentes módulos de la aplicación así como el modelo de contexto. El entorno visual que ofrece es un simulador de espacios 2D donde poder probar los desarrollos. Es aquí donde tiene cabida la participación de usuarios no técnicos, aunque esta participación se limita a la fase de validación y pruebas de la aplicación implementada por el programador.

Por su parte, el entorno OPEN, aunque pensado para usuarios no expertos en programación, tiene ciertas limitaciones. Por una parte, el modelo de datos basado en ontologías está pensado para ser diseñado por expertos en semántica y no ofrece mecanismos de extensión del modelo de una manera dinámica. Aunque esté pensado para un trabajo colaborativo entre usuarios programadores y no programadores, no ofrece una metodología bien definida para posibilitar esta colaboración. Además, las acciones que pueden ser desencadenadas por las situaciones identificadas están fijadas con anterioridad y no pueden ser extendidas.

En cuanto al acceso web a los entornos de desarrollo para su control y configuración, solamente OPEN lo contempla. El resto de entornos son en su mayoría librerías software que se configuran de manera programática. Este hecho dificulta por lo tanto la rápida adecuación del comportamiento de los sistemas sensibles al contexto. Un aspecto importante si se tiene en cuenta el entorno tan cambiante en el que se despliegan este tipo de sistemas

2.9. Metodologías de desarrollo software aplicadas a los sistemas sensibles al contexto

Además de contar con herramientas y entornos de desarrollo que permitan facilitar la programación de sistemas sensibles al contexto, es importante contar con metodologías de desarrollo adaptadas a este tipo de implementaciones. Los entornos de desarrollo y librerías software analizados anteriormente no contemplan ningún tipo de metodología que los usuarios de dichas herramientas puedan seguir para guiarse en la implementación de sistemas sensibles al contexto.

Por su parte, existen metodologías de desarrollo generales que pueden ser utilizadas en este tipo de implementaciones como son el *modelo de desarrollo en cascada*, el *iterativo* y el

modelo en espiral entre otros (Green y DiCaterino, 1998). Estas metodologías, aunque aplicables al desarrollo de sistemas sensibles al contexto, tienen una concepción de carácter general y no se centran en los aspectos más concretos involucrados en el desarrollo de este tipo de sistemas.

Con el fin de dotar de procesos de desarrollo adaptados a los sistemas sensibles al contexto, Henricksen e Indulska (2006) proponen una metodología de desarrollo, además de un lenguaje de modelado de información de contexto y un entorno de desarrollo en el que son aplicables. El lenguaje se denomina *Context Modelling Language* (CML) y está orientado a los diseñadores de este tipo de sistemas. La Figura 16 muestra un ejemplo de modelado utilizando los elementos proporcionados por CML, donde se muestran diferentes entidades de contexto y las relaciones entre las mismas.

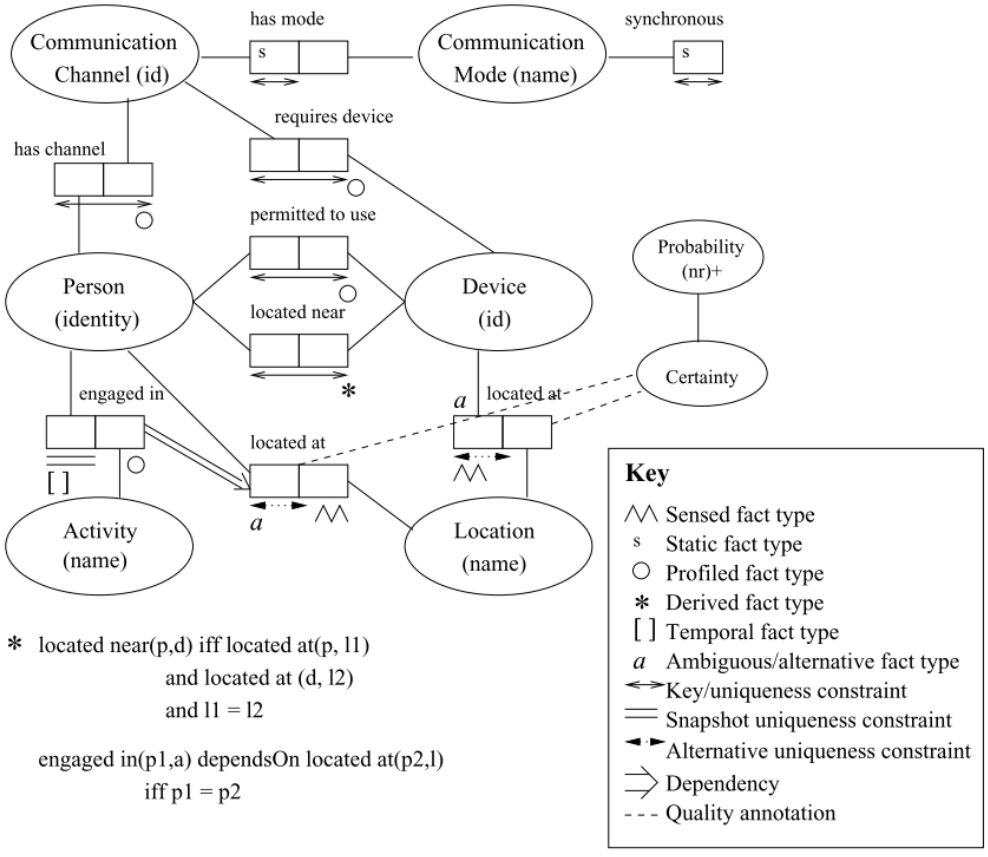


Fig. 16. Ejemplo de modelo de datos generado mediante CML (Henricksen e Indulska, 2006)

De esta manera, el lenguaje CML permite especificar los requerimientos de contexto que van a necesitar los sistemas a implementar. La metodología se divide en diferentes fases: análisis, diseño, implementación, adecuación de la infraestructura y validación. En las fases de análisis y diseño entra en juego el modelado mediante el lenguaje CML,

donde se definen las entidades de contexto necesarias, sus características, restricciones y relaciones entre las mismas. En la fase de implementación y adecuación de la infraestructura, se adapta el entorno de desarrollo proporcionado y se implementan sobre el mismo las diferentes funcionalidades requeridas en la fase de análisis.

Otra aproximación es la denominada como *Context-oriented Programming* (COP) (Hirschfeld et al., 2008). Esta aproximación, más que una metodología en sí misma, es una forma de programar, donde se contemplan mecanismos programáticos para que el software sea capaz de adaptarse a la información de contexto proveniente de los usuarios, del entorno y del propio sistema en el que se ejecutan.

Ambas metodologías están orientadas a los programadores de aplicaciones con lo que no contemplan la involucración de usuarios no técnicos o expertos en el dominio de aplicación, con el fin de poder definir y configurar de primera mano, la adaptabilidad de los comportamientos de los sistemas a implementar en función de la información de contexto.

2.10. Resumen

Desde la aparición de los primeros sistemas que utilizaban información de contexto a principios de los años 90 (Want et al., 1992), hasta hoy en día, se han sucedido numerosos avances en el campo de la computación sensible al contexto derivados de múltiples trabajos en los diversos ámbitos de la computación sensible al contexto (Hong et al., 2009). Aun así, y tal y como se ha ido analizando a lo largo de este capítulo, todavía quedan diferentes aspectos en los que no se ha establecido marco teórico alguno o no se han cubierto ciertas limitaciones y carencias para que este tipo de sistemas no sean meramente prototipos de laboratorio.

A continuación se resumen cada una de las carencias, necesidades y problemas encontrados en los trabajos revisados. En un primer bloque, se recogen las conclusiones relativas al análisis de la información de contexto. Un segundo apartado trata sobre los sistemas sensibles al contexto. Finalmente se resumen las carencias encontradas en el desarrollo de este tipo de sistemas.

2.10.1. Contexto y situación

No existe un marco común que defina lo que se entiende por contexto en un entorno computacional. Existen casi tantas definiciones de contexto como trabajos de

investigación realizados en torno a este concepto. Las definiciones analizadas pueden enmarcarse en tres tipos diferentes: aquellas que se sustentan en el entorno del usuario, las que contemplan la actividad del mismo o las que se centran en la situación del usuario. Por lo tanto, ninguna de estas definiciones es generalizable ni puede ser tomada como referencia ya que se centran en conceptos diferentes.

El único consenso existente en este ámbito es el haber tomado la definición propuesta por Dey (2001) como marco de referencia. Aun así, cabe destacar que esta definición tampoco puede ser establecida como marco general. Prueba de ello es que las entidades de contexto consideradas como relevantes en la citada definición, no contemplan entidades como los animales, los cuales pueden ser también entidades relevantes a tener en cuenta (Tang et al., 2005). Además, la definición se centra en la interacción del usuario con una aplicación, cuando puede haber escenarios en los que el usuario no realice ningún tipo de interacción con ninguna aplicación de manera explícita y sin embargo se produzca un proceso de adaptación en base a la información de contexto obtenida por el sistema. Ejemplos de este escenario se dan en los sistemas de Inteligencia Ambiental (Marzano y Aarts, 2003), donde es el entorno el que actúa de manera proactiva y autónoma como consecuencia de los datos de contexto obtenidos y procesados, sin que el usuario tenga la necesidad de interactuar previamente con ningún tipo de aplicación o dispositivo (Zhou et al., 2011). Por lo tanto, aunque es la definición de contexto más utilizada, tampoco puede ser considerada como una definición general de referencia sobre la que poder especificar otras definiciones más concretas y adaptadas a cada dominio de aplicación requerido.

En cuanto al término situación el panorama es idéntico. Diferentes autores han tratado de especificar lo que se entiende por situación, formulando aproximaciones dispares, desde las muy ambiguas (McCarthy y Hayes, 1968) hasta las más concretas que solamente pueden ser aplicadas en dominios muy particulares (Yau y Huang, 2006). Un hecho que es relevante y que se puede extraer de las distintas definiciones analizadas, es que la situación está en un nivel de abstracción por encima de la información de contexto. Por lo tanto, la situación de cualquier entidad es dependiente y viene determinada por la información de contexto.

Por su parte, la tipología de información que compone una situación puede ser también de lo más variada (Schmidt y van Laerhoven, 2001). De todos los tipos de información de contexto contemplados, se pueden determinar cuatro tipos que se

consideran primarios: la localización e identidad de las entidades de contexto, el tiempo (fecha y hora) y las condiciones ambientales. Así, aunque dependerá del dominio de aplicación en el que se vaya a implantar un sistema sensible al contexto, se deberá prestar especial atención a estos tipos de información de contexto.

Además, existen tantas categorías de información de contexto, como tipos de información diferentes se puedan contemplar. En base a las clasificaciones analizadas, puede establecerse una clasificación general compuesta por cuatro grupos: información referente al usuario, información del entorno físico, información sobre los dispositivos y el tiempo. En cada grupo se pueden enmarcar a su vez diferentes tipos de información de contexto.

En definitiva, no existe un marco común establecido y aprobado por la comunidad científica en relación a la información de contexto, sino que cada autor propone su marco conceptual adaptado al dominio de aplicación que se trate en cada trabajo de investigación.

En cuanto a la diversidad en la tipología y clasificación de contexto, se han podido extraer unas bases comunes a todos los trabajos analizados, pero su aplicación a nivel general dependerá del dominio de aplicación concreto y los requerimientos del mismo. Aun así, puede servir como referencia para todos aquellos noveles en el campo de la computación sensible al contexto.

2.10.2. Sistemas sensibles al contexto

Los diferentes sistemas sensibles al contexto implementados en los últimos años siguen una arquitectura generalizada, que se divide en cinco capas lógicas, cada una ofreciendo una serie de funcionalidades concretas que abstraen sobre las capas superiores de la arquitectura. Así, el esquema general de este tipo de sistemas se divide en capa de sensores, capas de obtención, procesamiento y gestión de datos de contexto y finalmente la capa de aplicación que hace uso de la información de contexto procesada en capas inferiores (Baldauf et al., 2006). Por ello, parece claro que esta arquitectura es válida y puede considerarse como marco de referencia en cualquier desarrollo en el que se utilice información de contexto.

Un aspecto clave en la arquitectura establecida es el lenguaje utilizado para modelar la información de contexto ya que condiciona la implementación de la capa de gestión de datos. Aunque en los últimos años predominan los sistemas cuyo modelo de datos se

ha implementado en base a ontologías (Chen, 2004; Sadeh, 2006), cualquier lenguaje de modelado es igualmente válido. Todo dependerá del grado de expresividad que requiera el modelo de datos en base a los requerimientos del sistema a implementar.

2.10.3. Desarrollo de sistemas sensibles al contexto

Uno de los factores importantes para el avance en el campo de los sistemas sensibles al contexto es posibilitar su desarrollo de una manera simple. Por lo tanto, tener herramientas que soporten la implementación de sistemas que hagan uso de información de contexto es esencial para que los programadores puedan crear nuevos servicios de una manera más sencilla, rápida, fiable y robusta.

Se han analizado diferentes propuestas de entornos y librerías de programación, pero todavía existen carencias en los mismos. Por una parte, estos entornos carecen de funcionalidades como la gestión automática de la información de contexto, teniendo el propio programador que gestionar las actualizaciones del modelo de datos de manera manual. Además, ninguno ofrece la posibilidad de extender el modelo de datos de manera dinámica ni de modificar los comportamientos definidos en función de los requerimientos precisados en cada momento. Estas limitaciones son críticas, ya que este tipo de sistemas se despliegan en entornos muy dinámicos y cambiantes, cuyos requerimientos pueden alterarse en función de nuevas fuentes de contexto que se quieran tener en cuenta y comportamientos nuevos que se quieran definir en función de la información de contexto disponible. Ninguno de los trabajos analizados ofrece herramientas para la gestión integrada de la localización de las entidades controladas por el sistema, ya que no incorporan sistemas de información geoespacial. Siendo la localización de las entidades uno de los tipos de información de contexto primarios, es vital que se ofrezcan mecanismos para su gestión.

Además, estas herramientas deben posibilitar la participación en el proceso de desarrollo de personal no técnico que pueda aportar su experiencia en el campo de aplicación concreto en el que se quiera implantar el sistema. Concretamente, la definición de las situaciones y los comportamientos derivados de tales situaciones es donde los expertos en el dominio de aplicación pueden contribuir de manera notable ya que conocen de primera mano las necesidades de los usuarios finales. A lo largo de la historia, diversos trabajos ha tratado de contrastar esta idea mediante prototipos sencillos (Sohn y Dey, 2003; Gajos et al., 2002; Zhang y Brügge, 2004), que aunque no

cuentan con todas las funcionalidades deseables para soportar el desarrollo de sistemas sensibles al contexto, han servido para evaluar su propósito con resultados esperanzadores. Por su parte, ninguna de las herramientas analizadas está pensada para la participación y colaboración de personal no técnico en el proceso de desarrollo, limitándose a ofrecer APIs de alto nivel que solamente los programadores son capaces de utilizar. Además, los entornos analizados necesitan de herramientas de programación para poder modificarlos por lo que no ofrecen ningún entorno de programación visual y accesible desde cualquier tipo de dispositivo y a través de la web.

Además de tener librerías y entornos de desarrollo adecuados, hay que disponer de metodologías de desarrollo que aprovechen las herramientas disponibles y posibiliten un ecosistema de colaboración entre programadores y expertos del dominio, para que puedan contribuir conjuntamente en el proceso de desarrollo de sistemas sensibles al contexto. Aunque se han encontrado metodologías de desarrollo orientadas a los sistemas sensibles al contexto (Henricksen e Indulska, 2006; Hirschfeld et al., 2008), están focalizadas para que sean empleadas por programadores, sin tener en cuenta al personal no técnico.

En resumen, no existen herramientas que cumplan los requerimientos deseables para soportar el desarrollo de sistemas sensibles al contexto, ni metodologías de desarrollo adaptadas que contemplen la colaboración de personas no técnicas (expertos en el dominio) en el proceso de implementación y configuración de este tipo de sistemas. Este aspecto es una carencia notable en el campo de la computación contextual ya que sin herramientas ni metodologías adaptadas a este campo, los sistemas desarrollados serán más costosos de implementar y más propensos a errores, lo que puede repercutir en la calidad del producto final y en la experiencia final del usuario.

Capítulo 3 Marco teórico

"No se puede desatar un nudo sin saber cómo está hecho"
Aristóteles (384 AC - 322 AC). Filósofo griego.

Partiendo del estado del arte analizado anteriormente en el ámbito de los sistemas sensibles al contexto, se ha establecido un marco teórico sobre el que se sustenta el presente trabajo de investigación. Se recogen desde la definición de conceptos fundamentales como son los de contexto y situación, hasta conceptos relacionados con la metodología de desarrollo diseñada descrita en el Capítulo 5 y los fundamentos de la plataforma para el desarrollo de sistemas sensibles al contexto descrita en el Capítulo 6.

3.1. Definiciones: contexto y situación

Como se ha visto en el estado del arte, existen diferentes aproximaciones a las definiciones de contexto y situación, en función de los autores y del trabajo realizado por los mismos. Para el presente trabajo se propone, en primer lugar, una definición de contexto general que posteriormente se particulariza en el ámbito de los sistemas sensibles al contexto. Como punto de partida de esta definición general se ha tomado la definición establecida por Dey (2001), la cual es la más referenciada en la literatura de este ámbito. De esta manera, a nivel general, el contexto se define del siguiente modo.

Contexto es cualquier información que resulta relevante para identificar la situación de una entidad.

Esta definición general establecida, se fundamenta también en las diferentes definiciones analizadas en el estado del arte (Sección 2.1.2), donde se establece que la situación está en un nivel de abstracción superior al de la información de contexto, y por lo tanto, es dependiente de esta.

Partiendo de esta definición general de contexto, se propone una definición más concreta y aplicada al marco del presente trabajo de investigación, donde se realiza una

aproximación al contexto orientada a los sistemas computacionales. De esta forma, se describe el contexto como sigue a continuación.

Contexto es cualquier información que resulta relevante para identificar la situación de una entidad y que puede ser obtenida y procesada por un sistema hardware o software con el fin de adaptar el comportamiento del mismo a la situación identificada.

Al igual que ocurre con la definición general de contexto propuesta, esta definición es también una adaptación de la enunciada por Dey (2001). La adaptación viene de la particularización de la misma en dos puntos fundamentales. Por un lado, se contempla que la información de contexto tiene que ser obtenida y procesada por un sistema hardware o software, en relación a la aproximación que se lleva a cabo a la noción de contexto desde los sistemas de computación. Por su parte, se concreta también la finalidad de este proceso de captación y procesamiento de la información de contexto referentes a una entidad por parte de los sistemas computacionales, que no es otra que la de obtener la situación de alto nivel y adaptar el comportamiento o funcionalidades del sistema a dicha situación. Esta finalidad se fundamenta en la Sección 2.1.3, donde se establece el carácter adaptativo de los sistemas sensibles al contexto.

Por lo tanto, la información de contexto debe poder ser obtenida y procesada por un sistema computacional, con el fin de adaptar su comportamiento en beneficio de la entidad que utilice el sistema, tanto mediante interacción directa con el mismo (pulsar un botón, realizar una búsqueda, etc.) como indirecta.

Existen elementos comunes en ambas definiciones como son la entidad y la situación de dicha entidad. En primer lugar, por *entidad* se establece lo siguiente.

Una entidad es cualquier ser vivo, objeto o lugar.

Así, se extiende el concepto de entidad propuesto por Dey (2001), teniendo cabida no solamente las personas, sino cualquier ser vivo en general, además de los objetos y lugares. De esta manera, ejemplos de entidad pueden ser una persona, un grupo de personas, un animal, un hotel, un dispositivo móvil, etc.

Por su parte, el concepto de *situación* se define de la siguiente manera.

Una situación es el estado de la información de contexto relevante en un punto o región en el espacio determinado, en un instante o intervalo en el tiempo concreto.

En esta definición se refleja también el nivel superior de abstracción que tiene una situación respecto a la información de contexto sobre la cual se define. Por lo tanto, el estado de la información de contexto de una entidad es la que determina la situación en la que se encuentra la misma. Esta definición tiene como referencia también la definición propuesta por Dey (2001), pero se extiende para tener en cuenta de manera explícita, dos parámetros de contexto fundamentales a la hora de definir una situación, como son el tiempo y el espacio.

De esta manera, el tiempo y el espacio resultan parámetros imprescindibles a la hora de definir e identificar una situación, por lo que se hacen necesarias referencias explícitas que informen de dónde y cuándo suceden las situaciones a detectar (Cook et al., 2007).

Las situaciones, además de ser dependientes de la información de contexto que se obtiene y procesa, pueden ser también dependientes o estar relacionadas con otras situaciones (Ye et al., 2011). A continuación se describen las diferentes relaciones posibles entre situaciones.

- *Generalización*: una situación puede considerarse más general que otra. Por ejemplo, la situación "viendo la televisión" es más específica que la situación "entretenimiento".

- *Composición*: una situación puede estar compuesta por diferentes situaciones de menor entidad. La situación "cocinando" está compuesta por otras como "utilizando un cuchillo".

- *Dependencia*: una situación puede depender de otra. Por ejemplo, la situación "volviendo del trabajo" es dependiente de "yendo al trabajo", es decir, que una no puede darse sin que previamente se haya dado la otra.

- *Contradicción*: dos situaciones pueden ser consideradas como mutuamente excluyentes si no pueden darse al mismo tiempo. Por ejemplo, las situaciones "cocinando" y "en la ducha" no pueden darse al mismo tiempo, por lo que son mutuamente excluyentes.

- *Relación temporal*: entre diferentes situaciones puede haber una correlación temporal, por lo que una situación puede ocurrir antes, después, etc. que otra. Por ejemplo, la situación "en el trabajo" se da después de "saliendo de casa". Es por lo tanto necesario contemplar otras situaciones que tienen

relación temporal directa con la situación que se quiere identificar en un instante determinado. Las relaciones temporales que se pueden dar entre situaciones se basan en la lógica temporal de Allen (1983), las cuales se reflejan en la siguiente tabla.

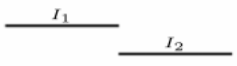
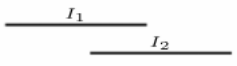
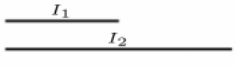
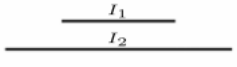
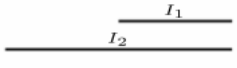
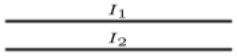
Relation	*Illustration*	*Interpretation*
$I_1 < I_2$ $I_2 > I_1$		I_1 before I_2 I_2 after I_1
$I_1 \mathsf{m} I_2$ $I_2 \mathsf{mi} I_1$		I_1 meets I_2 I_2 met by I_1
$I_1 \mathsf{o} I_2$ $I_2 \mathsf{oi} I_1$		I_1 overlaps I_2 I_2 overlapped by I_1
$I_1 \mathsf{s} I_2$ $I_2 \mathsf{si} I_1$		I_1 starts I_2 I_2 started by I_1
$I_1 \mathsf{d} I_2$ $I_2 \mathsf{di} I_1$		I_1 during I_2 I_2 contains I_1
$I_1 \mathsf{f} I_2$ $I_2 \mathsf{fi} I_1$		I_1 finishes I_2 I_2 finished by I_1
$I_1 = I_2$		I_1 equals I_2

Fig. 17. Relaciones temporales basadas en la lógica temporal de Allen (Guesgen, H. W., y Marsland, S., 2010).

En cuanto a la clasificación de la información de contexto que compone una situación, se establecen dos niveles: información de bajo nivel e información de alto nivel. La información de bajo nivel es aquella que no ha sido procesada por ningún sistema, mientras que la información de alto nivel es la que resulta del procesamiento de la información de bajo nivel (Guan et al., 2007). Por lo tanto, las situaciones se consideran como información de alto nivel.

A su vez, se establece una clasificación de la información de contexto de bajo nivel en estática y dinámica. La información estática es la que no varía en el tiempo, como por ejemplo la fecha de nacimiento de una persona. La información dinámica es aquella que cambia constantemente a lo largo del tiempo, como por ejemplo, la temperatura de una

habitación. La información dinámica es la que más comúnmente tiene que ser gestionada en este tipo de sistemas, por lo que requiere de especial atención a la hora de implementar un sistema sensible al contexto.

Para explicar de una manera visual todos estos conceptos y características relativas a la información de contexto, se presenta la siguiente figura donde se plasman los conceptos definidos y algunos ejemplos de relaciones entre los mismos.

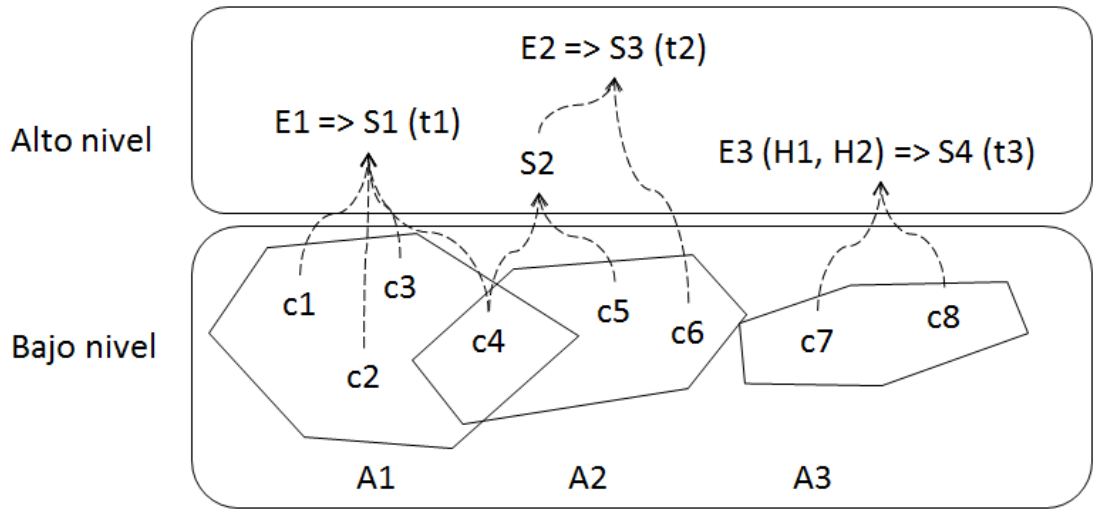


Fig. 18. Representación gráfica de los conceptos de *contexto* y *situación*

Por una parte, se tiene un conjunto de diferentes tipos de información de contexto (c1, c2, c3, c4, c5, c6, c7 y c8), que pueden ser tanto información estática como dinámica. Esta información de contexto puede ser obtenida y procesada por un sistema computacional con el fin de identificar las diferentes situaciones (S1, S2, S3 y S4) relativas a las diferentes entidades representadas (E1, E2 y E3). Toda esta información de contexto de bajo nivel que el sistema es capaz de procesar, puede ser agrupada en diferentes subconjuntos en base a distintas regiones en el espacio, representadas mediante áreas (A1, A2 y A3) donde las situaciones a identificar son relevantes en un intervalo o instante de tiempo determinado (t1, t2 y t3). Existe además la posibilidad de que una situación (S3) pueda estar determinada por información de bajo nivel y por información de alto nivel (S2) en base a la relación de composición entre situaciones. Además, una situación (S4) puede tener una relación temporal con situaciones anteriores en las que se encontraba una entidad determinada y se identifica gracias a información de bajo nivel y al histórico de situaciones anteriores (H1 y H2) de la entidad (E3).

De esta manera, se pueden configurar diferentes subconjuntos de información de contexto en base a determinadas áreas en las cuales dicha información tiene un peso relevante en la identificación de determinadas situaciones. Un ejemplo sería el establecido por la situación "esperando al bus" que se representa en la siguiente figura.

Fig. 19. Relación entre una situación y una región en el espacio

Esta situación solamente puede darse en una parada de autobús, por lo que la información de contexto que se utiliza para identificar dicha situación, está relacionada con el área o región del espacio referente a la parada de bus. En este caso concreto, podrían ser suficientes los datos de la velocidad del usuario y su localización, para identificar la situación en esa región concreta.

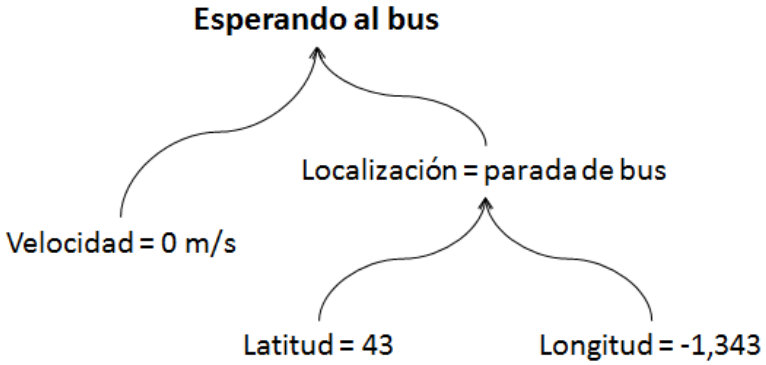


Fig. 20. Información de bajo nivel necesaria para identificar la situación "esperando al bus"

Sin embargo, esas mismas condiciones de la información de contexto en una región del espacio diferente pueden ser identificadas como una situación totalmente diferente. De esta manera, se establece una estrecha relación entre la región en el espacio y las situaciones que en ella pueden ser identificadas.

Esta aproximación basada en agrupamiento de datos de contexto en base a regiones en el espacio en las que determinadas situaciones tienen relevancia, se fundamenta

también en la primera ley de la geografía enunciada por Waldo Tobler (1970), donde se especifica que *todas las cosas están relacionadas entre sí, pero las cosas más próximas en el espacio tienen una relación mayor que las distantes*[29]. Haciendo una interpretación y aplicación a la presente definición de situación, se puede decir que tienen mayor relevancia en la detección de una situación aquellos valores de contexto que estén asociados a una región del espacio determinada. De esta manera, se acotan y priorizan los tipos de información de contexto más significativos en función de la región del espacio en la que se encuentra una entidad.

3.2. Metodología de desarrollo

En esta sección se establece el marco teórico que fundamenta la metodología de desarrollo para la creación de sistemas sensibles al contexto. Así, resulta vital contar con un proceso de desarrollo guiado por metodologías de desarrollo orientadas a este ámbito. Como se ha visto en el análisis del estado del arte, existen muy pocas metodologías de desarrollo centradas en los sistemas sensibles al contexto (Henricksen e Indulska, 2006; Hirschfeld et al., 2008). Además, las metodologías revisadas están orientadas a usuarios técnicos con conocimientos de programación, por lo que imposibilitan la involucración de expertos en el dominio.

En este caso, el marco teórico utilizado para la definición de la metodología que se plantea para guiar el proceso de desarrollo de sistemas sensibles al contexto mediante entornos de desarrollo orientados a la construcción de este tipo de sistemas, puede descomponerse en dos requisitos fundamentales de la misma (RM#), los cuales se describen en las siguientes secciones.

3.2.1. RM1 - Centrada en la definición de situaciones

Uno de los puntos clave en el desarrollo de sistemas sensibles al contexto es la detección de situaciones que resulten relevantes para el funcionamiento del sistema. Por ello, la metodología debe centrarse en la definición de las situaciones que resultan relevantes y que el propio sistema tiene que ser capaz de detectar con el fin de adaptar su comportamiento a las mismas. Para ello, se establece que la metodología debe incluir un proceso de especificación de aquellos parámetros referentes a las diferentes situaciones a ser consideradas con el fin de guiar el desarrollo del sistema a posteriori.

[29] "*Everything is related to everything else, but near things are more related to each other"* (Tobler, 1970).

Se establece así que esta fase de especificación de la metodología debe basarse en dar respuesta a cinco preguntas fundamentales sobre las situaciones, que se fundamentan en los conceptos establecidos por Oh et al. (2006), denominados 4W1H, aplicados a la propia definición de una situación. A continuación se listan las cuestiones derivadas de cada uno de estos cinco conceptos.

1. ¿Qué situación es? *(What)*
2. ¿Quién puede encontrarse en la situación? *(Who)*
3. ¿Dónde se produce la situación *(Where)*
4. ¿Cuándo se produce la situación? *(When)*
5. ¿Cómo se detecta la situación? *(How)*

Las diferentes respuestas a estas preguntas tienen relación directa con las definiciones establecidas en la Sección 3.1 sobre contexto y situación. Así, la pregunta 1 hace referencia a la identificación de la propia situación mediante un nombre. La pregunta 2 hace referencia a las entidades que pueden encontrarse en dicha situación. La pregunta 3 hace referencia a la noción espacial de la situación. Por su parte, la pregunta 4 hace referencia a la noción temporal de la situación. Finalmente, la pregunta 5 hace referencia a la información de contexto necesaria para la detección de la situación.

De esta manera se cubren los aspectos que constituyen una situación tal y como se recoge en la definición establecida para el presente trabajo de investigación. Estos aspectos deben poder trasladarse además de manera sencilla a la herramienta de desarrollo utilizada para la implementación de sistemas sensibles al contexto.

3.2.2. RM2 - Colaborativa

La metodología debe impulsar y facilitar la colaboración entre expertos en el dominio y programadores, con el fin de que la especificación de las situaciones pueda complementarse con los conocimientos técnicos del personal programador y los conocimientos en el dominio de aplicación proporcionados por los expertos en el dominio.

Existen metodologías de desarrollo software que se basan en la colaboración de diferentes actores dentro del proceso de desarrollo cuyos principios pueden ser de aplicación a la metodología. Concretamente, la metodología de desarrollo software denominada como Programación Extrema o *Extreme Programming (XP)* formulada por

Kent Bleck (2004) tiene entre sus pilares fundamentales este principio de colaboración. Esta metodología se considera como un proceso de desarrollo ágil, ya que permite adaptar el desarrollo de un sistema software a los requerimientos variantes en el tiempo en el mismo momento en el que se producen estos cambios. El objetivo de la metodología es hacer un software de calidad y de la forma más rápida posible.

Entre las características de esta metodología destaca la que hace referencia a la comunicación, fruto de la conceptualización de un trabajo de desarrollo que se realiza en parejas (Programación en Pareja o *Pair Programming*). Al estar dos personas realizando labores de implementación, se supone una mayor calidad del trabajo realizado ya que está en continua supervisión por ambas partes además de dar lugar a una comunicación continua entre ambas personas.

Además, la metodología integra también al cliente final en el propio proceso de desarrollo, haciéndole partícipe en todo momento del producto a desarrollar. Esta característica encaja a la perfección con la inclusión de los expertos en el dominio de aplicación en la metodología para el desarrollo de sistemas sensibles al contexto que se plantea. Así, en todo momento el experto en el dominio tendrá la percepción de si lo que se está desarrollando corresponde con los requerimientos del sistema establecidos en un principio.

Por lo tanto, este concepto de colaboración en el proceso de desarrollo tiene que estar presente en la metodología como pilar fundamental en el ciclo de vida del desarrollo de sistemas sensibles al contexto.

3.3. Plataforma para el desarrollo de sistemas sensibles al contexto

En esta sección se tratan los fundamentos teóricos establecidos para la definición de la plataforma para el desarrollo de sistemas sensibles al contexto. Así, en base al análisis del estado del arte y las carencias encontradas en los entornos para el desarrollo de sistemas sensibles al contexto y teniendo en cuenta las definiciones propuestas de contexto y situación, se han establecido una serie de premisas, características y funcionalidades que conforman el marco teórico sobre el que se define la plataforma de desarrollo. Este marco teórico deriva en una serie de requisitos que la plataforma de desarrollo debe cumplir con el fin de hacer frente a las carencias encontradas en la revisión del estado del arte.

A continuación, se describen cada uno de los requisitos de primer nivel identificados (RP#), empezando por la arquitectura de la plataforma y siguiendo con cada una de las características funcionales que debe cumplir para cubrir las carencias encontradas en otros entornos de desarrollo de sistemas sensibles al contexto. A su vez, por cada requisito de primer nivel, se identifican una serie de requisitos de segundo nivel en los que se descomponen (RP#.#).

3.3.1. RP1 - Arquitectura de la plataforma

La arquitectura de la plataforma debe contar con las capas ya identificadas en el estado del arte y que la gran mayoría de sistemas sensibles al contexto y entornos de desarrollo siguen (Baldauf y Dustdar, 2006). Estas capas abstraen las diferentes funcionalidades que este tipo de entornos ofrecen simplificando su configuración.

Existen diferentes ámbitos de investigación que están estrechamente relacionados con los sistemas sensibles al contexto que pueden ser de aplicación para enriquecer la arquitectura de los mismos. En el presente trabajo se ha tenido en cuenta una aproximación de convergencia científico-tecnológica a partir de disciplinas complementarias que pueden enriquecer la concepción y adopción de la arquitectura de los sistemas y entornos de desarrollo de los sistemas sensibles al contexto, como son la Web de las Cosas, Desarrollo Orientado al Usuario Final, Computación en la Nube y Sistemas Reactivos.

RP1.1 Web de las Cosas. El ámbito denominado Web de las Cosas o *Web of Things (WoT)* (Guinard et al., 2011) hace referencia a una visión derivada del ámbito conocido como Internet de las cosas o *Internet of Things*, donde objetos cotidianos están conectados a la Web y su información y funcionalidad es accesible mediante los protocolos ya establecidos para la misma (*Hypertext Transfer Protocol* - HTTP). Esta disciplina ha establecido una serie de buenas prácticas en cuanto a la definición de funcionalidades e información que los objetos deben exponer y la manera de acceder a las mismas. Estas buenas prácticas o guías pueden ser aplicadas en la capa de sensores y en la capa de acceso a datos que componen la arquitectura de los sistemas sensibles al contexto. De esta manera los sensores se modelarían siguiendo estas pautas, siendo accesibles a través de la Web. Se consigue así un acceso más sencillo y homogéneo a la información de contexto, consiguiendo una abstracción de las funcionalidades y

software/hardware específico de cada fuente de contexto y simplificando por lo tanto el desarrollo de los sistemas sensibles al contexto.

De esta manera, la aplicación de los conceptos relativos al ámbito de la Web de las Cosas, llevan al establecimiento para el presente trabajo de investigación, de la siguiente premisa fundamental.

Las fuentes que proporcionan información de contexto o de bajo nivel, siguen las buenas prácticas establecidas en el ámbito de la Web de las Cosas.

Esta premisa hace referencia a que las diversas fuentes de datos que pueden proporcionar información de contexto (p. ej. dispositivo móvil, sensor y servicio web) deben proporcionar interfaces de acceso Web basados en protocolos HTTP con los que poder acceder a los diferentes datos de contexto proporcionados por las fuentes.

RP1.2 Desarrollo Orientado al Usuario Final. Otro de los ámbitos de conocimiento detectados que pueden contribuir a propiciar la orientación de este tipo de entornos hacia usuarios no expertos en programación, como es el caso de los expertos en el dominio, es el Desarrollo Orientado al Usuario Final o *End-User Development (EUD).* El desarrollo orientado al usuario final se refiere al *conjunto de métodos, técnicas y herramientas que capacitan a usuarios no programadores a modificar y extender los sistemas software que manejan*[30] (Lieberman, 2006).

La aplicación de este concepto a los entornos de desarrollo de sistemas sensibles al contexto posibilita que usuarios no técnicos, pero expertos en el dominio de aplicación, puedan participar en el proceso de desarrollo facilitando así la labor a los desarrolladores y mejorando el proceso de desarrollo y la calidad del producto final. Belloti y Edwards (2001) proponen que los usuarios deben conocer cómo el sistema interpreta la información de contexto y este proceso debe ser comprensible para los mismos. Además debe proporcionar mecanismos para que los usuarios puedan adaptar el comportamiento de los sistemas, tengan o no conocimientos de programación. Trigg et

[30] "*End-User Development can be defined as a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact*" (Lieberman, 2006).

al. (1987) establecen que un *sistema es adaptable si un usuario final puede producir nuevos comportamientos del sistema sin necesidad de los programadores o diseñadores del propio sistema*[31].

RP1.3 Computación en la Nube. Por su parte, una de las tendencias actuales en los paradigmas de computación que pueden tener aplicación en el despliegue de este tipo de entornos de desarrollo, es la denominada Computación en la Nube o *Cloud Computing* (Hayes, 2008). Esta tendencia se basa en el alojamiento externo de la propia plataforma de desarrollo. De esta manera, los recursos hardware y el mantenimiento que necesita la plataforma quedarían gestionados por terceros, acelerando la implantación y adopción de la misma por parte de los usuarios.

Por lo tanto, la arquitectura de referencia puede ser enriquecida mediante la aplicación de conceptos, avances alcanzados y tendencias en estas disciplinas, las cuales se alinean con los objetivos iniciales establecidos en este trabajo de investigación.

En la siguiente figura se muestra en qué capas se propone que complementen las directrices anteriores la arquitectura de los entornos para el desarrollo de sistemas sensibles al contexto.

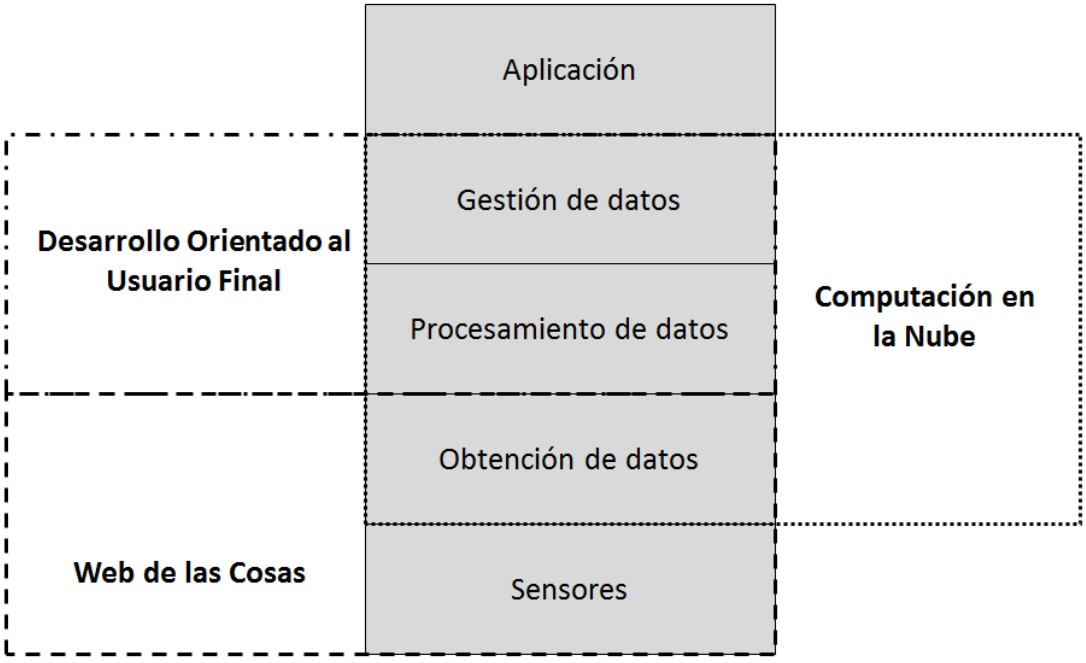


Fig. 21. Arquitectura para sistemas sensibles al contexto influenciada por campos científico-tecnológicos complementarios

[31] *"A system is adaptable if an end-user produces new system behaviour without help from programmers or designers"* (Trigg et al., 1987).

Como puede observarse, sobre la arquitectura de referencia establecida por Baldauf y Dustdar (2006) se han conectado capas referentes a los conceptos tratados anteriormente. Por una parte, los principios establecidos en la Web de las Cosas pueden ser de aplicación en capas inferiores de la arquitectura, donde residen las fuentes de contexto y los mecanismos de obtención de datos de dichas fuentes. Como ya se ha comentado anteriormente, esta nueva capa basada en *WoT* proporciona un acceso homogéneo a todas las fuentes de contexto necesarias por el sistema, simplificando así el proceso de obtención de información de contexto.

Las capas superiores relativas a los procesos de gestión de la información de contexto son las que están más directamente ligadas a los principios establecidos por el Desarrollo Orientado al Usuario Final. De esta manera, las funcionalidades de procesamiento y gestión de datos de contexto deben posibilitar la involucración de personal no técnico en su configuración.

Finalmente, las capas responsables de la gestión de la información de contexto (gestión, procesamiento y obtención de datos) pueden ser desplegadas en una infraestructura en nube, ofreciendo así una serie de servicios a las capas de aplicación y sensores. Estos servicios expuestos, posibilitarán la gestión de la información de contexto.

RP1.4 Carácter reactivo y adaptativo. Por su parte, y atendiendo a los patrones de computación analizados con anterioridad (Gwizdka, 2000), se propone realizar una aproximación al carácter reactivo y adaptativo que este tipo de sistemas tiene. Esta aproximación se fundamenta en la propia definición de contexto establecida en el presente trabajo, donde se establece que la información de contexto se obtiene y procesa con el fin de identificar la situación de las entidades relevantes para el sistema. Por lo tanto, el sistema debe reaccionar y adaptarse a las situaciones detectadas. Este comportamiento se basa además en la definición propuesta por Harel y Pnueli (1985) sobre los *sistemas reactivos*, que establecen que son aquellos *sistemas que deben tratar los eventos recibidos de su entorno y reaccionar en consecuencia*[32].

Se considera además, que la adaptación del sistema a la situación detectada debe realizarse de manera independiente, sin que el usuario final tenga la necesidad de realizar

[32] *"Reactive systems are repeatedly prompted by the outside world and their role is to continuously respond to external inputs"* (Harel y Pnueli, 1985).

ningún tipo de interacción directa con el sistema para que éste adapte su comportamiento. Así, se establece que en un entorno de computación contextual, es el propio sistema el que tiene que "reaccionar" en base a los datos de contexto procesados y la situación detectada, mejorando así la interacción entre el usuario y el sistema (Lamsfus, 2010), ya que es el propio sistema el que se anticipa a las necesidades de los usuarios sin intervención explícita de los mismos, convirtiéndose en un sistema adaptativo (Conlan et al., 2003).

Si bien los diferentes patrones establecidos en el estado del arte son válidos (Sección 2.4), se propone que una combinación de los mismos es más adecuada para ser aplicada a los sistemas sensibles al contexto. De esta manera, en el patrón propuesto se combinan como entrada al sistema, los datos de contexto, los datos del usuario (como datos introducidos en el sistema, etc.) y reglas que sirven para modelar las situaciones en base a los datos de contexto y los datos introducidos por el usuario (si los hubiera). En función de estas reglas de comportamiento, se producen unas salidas con información de alto nivel referente a la situación detectada. En la siguiente figura se refleja el resultado de la combinación resultante de los dos patrones de computación revisados en el estado del arte que son de aplicación para el presente trabajo de investigación.

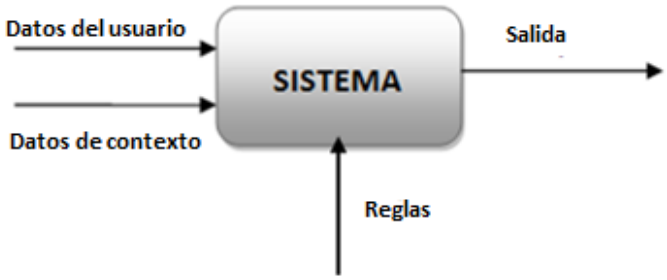


Fig. 22. Propuesta de patrón de computación para el procesamiento de datos de contexto

Teniendo en cuenta el patrón propuesto, se establece que la plataforma debe actuar a modo de elemento intermedio o *middleware*[33] (Bishop y Kame, 2003) entre los datos de

[33] Se entiende como *middleware* el software que asiste en la interacción con otros sistemas software y/o hardware, simplificando el trabajo de los programadores. Provee de herramientas para mejorar la calidad de servicio, seguridad, envío de mensajes, etc. "*Middleware is the software that assists an application to interact or communicate with other applications, networks, hardware, and/or operating systems. This software assists programmers by relieving them of complex connections needed in a distributed system. It provides tools for improving quality of service (QoS), security, message passing, directory services, file services, etc. that can be invisible to the user.*" (Bishop y Kame, 2003).

entrada de contexto y el sistema final a ser implementado, tal y como se muestra en la siguiente figura.

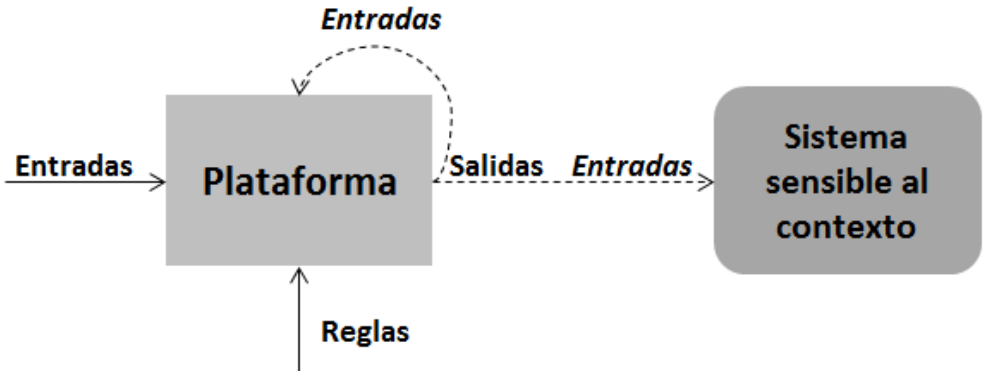


Fig. 23. Plataforma como elemento de procesamiento y transformación de información de contexto de bajo nivel a información de contexto de alto nivel

De esta manera, la plataforma de desarrollo es la que recibe los datos de entrada, que corresponderán a información de bajo nivel de contexto, y la que en base a una serie de reglas de comportamiento definidas, producirá salidas de alto nivel que a su vez servirán como entrada a los sistemas que quieran reaccionar y adaptarse a las situaciones detectadas. Las propias salidas de la plataforma pueden también actuar como entradas enriqueciendo así los datos de bajo nivel procesados. Así, se posibilita también el encadenamiento de reglas en base a las salidas generadas por las mismas.

Esta disposición facilita que la plataforma sea el elemento donde se aglutine y gestione la información de contexto, convirtiéndose en un módulo independiente a cualquier sistema sensible al contexto que se quiera desarrollar. Esto posibilita que el sistema final pueda estar implementado en cualquier lenguaje de programación. Además, facilita también la configuración y gestión de la información de contexto sin que el sistema sensible al contexto tenga que ser modificado, consiguiendo así una capa de abstracción entre la propia gestión de la información de contexto y el sistema o servicio final a implementar.

3.3.2. RP2 - Modelo de datos

La definición de un modelo de datos que sirva para representar y almacenar la información de contexto es fundamental para que los sistemas computacionales puedan procesarla. Este modelo de datos debe servir por lo tanto, para representar todos los datos de contexto que actúan a modo de entradas para su posterior gestión e

identificación de las situaciones requeridas en base a dichos datos. A continuación se detallan los tres requisitos específicos que debe cumplir el modelo de datos.

RP2.1 Requisitos generales. Existen una serie de condiciones generales que el modelo de datos debe cumplir (Bettini et al., 2010).

- *Heterogeneidad*: el modelo de datos debe posibilitar la representación de datos provenientes de diferentes fuentes de contexto. Estas fuentes de contexto proporcionan por lo general datos que tienen una naturaleza y tipología muy diferente entre sí, presentando diferentes formatos de representación y tipos de datos.

- *Dependencias y relaciones*: el modelo de datos debe posibilitar la representación de las relaciones de dependencia entre diferentes datos de contexto. Así, se establece que el modelo debe permitir, por una parte, la definición de entidades de contexto que aglutinen información en base a propiedades o atributos de la entidad, y que posibiliten además el establecimiento de relaciones con otras entidades de contexto mediante las propiedades o atributos definidos.

- *Inferencia*: el modelo debe posibilitar la aplicación de mecanismos de razonamiento para poder inferir nuevo conocimiento sobre los datos representados de manera explícita en el propio modelo de datos y transformar de esta manera información de bajo nivel en información de alto nivel.

- *Flexibilidad*: el modelo de datos debe ser libre, pudiendo establecer cualquier entidad y propiedad necesaria para la representación de datos de contexto.

RP2.2 Soporte para representación espacial. En los sistemas sensibles al contexto, donde la movilidad de las entidades es un factor determinante, uno de los parámetros fundamentales es la localización de las mismas. Es por ello que el modelo de datos debe posibilitar la representación de parámetros de localización como la latitud y la longitud de las entidades que puedan estar en movilidad o de las que se tenga especial interés en conocer su localización.

Además, y atendiendo a la definición de situación propuesta, se hace necesaria la inclusión de regiones espaciales donde se puedan detectar determinadas situaciones de las entidades relevantes para el sistema como parte del modelo de datos.

RP2.3 Soporte para representación temporal. Otro de los aspectos fundamentales, tal y como se refleja en la definición de situación, es proporcionar el soporte necesario para aplicar lógica temporal sobre el modelo de datos. Por ello, es importante que en el modelo se incluyan propiedades que puedan almacenar datos temporales (fecha y hora) con el fin de aplicar operadores temporales sobre las entidades definidas.

3.3.3. RP3 - Razonamiento

Como ya se ha establecido en uno de los requisitos generales del modelo de datos (RP2.1), se debe contemplar un mecanismo de razonamiento que pueda ser aplicado al modelo y sobre el que se pueda inferir información de alto nivel. De esta manera, el razonamiento estará orientado a la identificación de situaciones. El mecanismo de razonamiento debe cumplir una serie de requisitos que se describen a continuación.

RP3.1 Mecanismo basado en reglas. Como se recoge en el estado del arte (Sección 2.7) existen diferentes tipos de mecanismos o técnicas de razonamiento que se pueden utilizar para la inferencia de situaciones de alto nivel, como aquellas basadas en la especificación de reglas lógicas o los mecanismos basados en el aprendizaje.

Se propone utilizar un sistema basado en reglas, dado que los basados en el aprendizaje tienen el inconveniente de necesitar una gran cantidad de datos para configurar un modelo fiable con el que poder identificar situaciones. En un entorno de computación sensible al contexto, la información es muy variable y dinámica por lo que tener sistemas basados en el aprendizaje ralentizaría la implantación del sistema cada vez que hubiera algún tipo de cambio en las fuentes de contexto o en las situaciones a identificar. Además, ya que en este tipo de sistemas se tienen que especificar a priori las reglas que conforman o modelan, en este caso, las situaciones, los usuarios no técnicos pueden volcar todo su conocimiento como expertos en el dominio mediante reglas lógicas de comportamiento definidas en la propia plataforma y utilizando el modelo de datos definido.

Esta elección del mecanismo de razonamiento basado en reglas está alineada con la arquitectura de la plataforma propuesta anteriormente, donde las reglas de

comportamiento son una entrada más del sistema cuya finalidad es determinar las situaciones a ser detectadas en base a la información de contexto recibida.

RP3.2 Razonamiento espacial. Se debe posibilitar el razonamiento espacial sobre el modelo de datos definido. Es fundamental para el sistema conocer si una entidad está en una región del espacio concreta con el fin de identificar las situaciones que pueden darse en dicha región del espacio. De esta manera, el sistema de razonamiento puede centrarse en el procesamiento de la información de contexto asociada a regiones espaciales determinadas en las que se encuentran las entidades relevantes para el sistema que a su vez tengan modeladas en base a reglas lógicas las situaciones a ser detectadas.

Esto significa que las reglas se activarán solamente cuando una entidad se encuentre en una región del espacio concreta y en ella se hayan definido una serie de situaciones para la entidad que el sistema debe ser capaz de detectar. De esta forma se optimiza también el proceso de razonamiento ya que se acota a aquellas ocasiones en las que las entidades se encuentran en regiones espaciales definidas en el modelo de datos.

RP3.3 Razonamiento temporal. Por su parte, el razonamiento temporal es también fundamental, tal y como refleja la definición de situación anteriormente establecida. De esta manera, el sistema de razonamiento debe posibilitar la inferencia en base a la lógica temporal de Allen (1983).

3.3.4. RP4 - Gestión automática de la información de contexto

Un aspecto que debe contemplar la capa de gestión de la información de contexto es la automatización de la gestión de los datos de contexto. Por lo general, la gestión de la información de contexto conlleva programar funcionalidades concretas en cada nuevo desarrollo para poder posibilitar esta gestión. Se puede decir, que este código puede ser un tanto repetitivo y puede ser generalizado para ser utilizado en cualquier tipo de desarrollo en el que se necesite una gestión en los datos de contexto. Por lo tanto, la plataforma tiene que ser capaz de automatizar los siguientes procesos relacionados con la gestión de la información de contexto.

RP4.1 Transformación. La plataforma debe ser capaz de convertir la información de contexto al modelo de datos definido de manera automática en base a configuraciones realizadas por los usuarios de la plataforma.

RP4.2 Inserción y actualización. La plataforma debe ser capaz de insertar y actualizar las instancias de las entidades de contexto en base a los nuevos datos de contexto recibidos

de forma automática. De esta manera, si una determinada información de contexto no se encontrara almacenada previamente en la plataforma, se debería insertar en el módulo de gestión de la plataforma de manera automática. Si ya se encontrara en la plataforma, la información de contexto debería ser actualizada con el nuevo dato de manera automática.

RP4.3 Agregación. Se tienen que establecer mecanismos para fusionar la información de contexto proveniente de diferentes fuentes de contexto sobre las entidades definidas en el modelo de datos. De esta manera, diferentes datos de contexto que provienen de fuentes de contexto distintas pero que pertenecen a la misma instancia de entidad definida en el modelo de datos, deben poder ser agregados o unificados sobre la misma instancia por la plataforma. Un ejemplo de este comportamiento sería el dado por dos fuentes de contexto, como pudieran ser un dispositivo móvil con GPS y un pulsómetro, que proporcionaran información de contexto como la localización y el ritmo cardíaco respectivamente, asociados a una misma instancia de entidad de tipo "Persona". Al pertenecer ambos datos de contexto a la misma persona, los datos tendrían que ser unificados por la plataforma almacenándolos en la misma instancia de entidad que representa a la "Persona".

RP4.4 Recolección de basura. Resulta necesario que la plataforma pueda gestionar las instancias de entidades que ya no se utilizan, no son actualizadas o no son referenciadas en el sistema eliminándolas del mismo de manera automática. Este proceso puede optimizar el funcionamiento de la plataforma y el proceso de razonamiento ya que se reduce el número de instancias que almacena la plataforma.

3.3.5. RP5 - Extensible

Como se ha mencionado anteriormente, los sistemas que utilizan información de contexto son por naturaleza dinámicos: nuevos sensores pueden ser identificados como fuentes de contexto y/o nuevas situaciones pueden ser requeridas lo que puede significar la modificación del modelo de datos y las reglas definidas. Por lo tanto, un sistema de este tipo tiene que ser lo suficientemente flexible para ser extendido de manera dinámica y en tiempo real o de ejecución (Ye et al., 2011) afectando lo menos posible al conjunto general del sistema implementado.

RP5.1 Fuentes de contexto. La plataforma debe permitir la incorporación de nuevas fuentes de contexto para que sean gestionadas, además de poder eliminar o desactivar

las ya existentes, con el mínimo impacto sobre el resto de información gestionada por la misma.

RP5.2 Modelo de datos. La plataforma tiene que permitir extender el modelo de datos, pudiendo modificarlo en cualquier momento. Esta modificación implica la creación de nuevas entidades de contexto, la modificación de las ya existentes y el borrado de entidades anteriormente creadas. Además, las regiones espaciales que formen parte del modelo de datos deben poder ser también modificadas. De esta forma, el modelo puede ser adaptado a los nuevos requerimientos de información de contexto y fuentes de información pertinentes.

RP5.3 Reglas. Las reglas que modelan el comportamiento y las situaciones a detectar por el sistema de razonamiento deben ser extensibles, pudiendo modificar, insertar y eliminar reglas en tiempo de ejecución. Esto posibilita que los comportamientos definidos y las configuraciones realizadas para identificar situaciones puedan ser modificados en base a nuevos requerimientos.

Mediante la extensión de las fuentes de contexto, el modelo de datos y las reglas de comportamiento, los sistemas sensibles al contexto pueden ser flexibles y adaptarse a los requerimientos cambiantes propios de los entornos en los que se ejecutan.

3.3.6. RP6 - Movilidad

Como ya se menciona anteriormente, la localización es uno de los parámetros de contexto fundamentales a tener en cuenta en cualquier desarrollo de sistema sensible al contexto. Prueba de ello son los primeros sistemas considerados como sensibles al contexto, denominados Servicios Basados en la Localización o *Location Based Services (LBS)* (Steiniger et al., 2006), que contemplan la localización como parámetro fundamental de contexto para poder personalizar su comportamiento.

Es por ello, que el entorno para el desarrollo de sistemas sensibles al contexto debe proporcionar un soporte integrado para poder gestionar la movilidad de las diferentes entidades de contexto que constituyen el modelo de datos y las reglas de comportamiento definidas.

De esta manera, resulta de vital importancia contar con un Sistema de Información Geográfica (SIG) incorporado en la plataforma que sea capaz de gestionar la localización de las entidades del modelo de datos que precisen tal gestión de la movilidad y sobre el que se sustenten operaciones geoespaciales que las diferentes capas

de la plataforma puedan requerir. Un ejemplo de operación geoespacial de interés es aquella en la que se necesita conocer si una determinada entidad con movilidad se encuentra dentro de una región espacial determinada, con el fin de conocer si dicha entidad se encuentra en una de las situaciones modeladas en la plataforma y asociadas a dicha región.

La ubicación del módulo SIG dentro de la arquitectura de referencia vendría determinada por el módulo de Gestión de datos, pudiendo acceder a los datos de contexto gestionados por el mismo. En la siguiente figura se refleja el módulo SIG añadido a la arquitectura de referencia de la arquitectura de la plataforma para el desarrollo de sistemas sensibles al contexto.

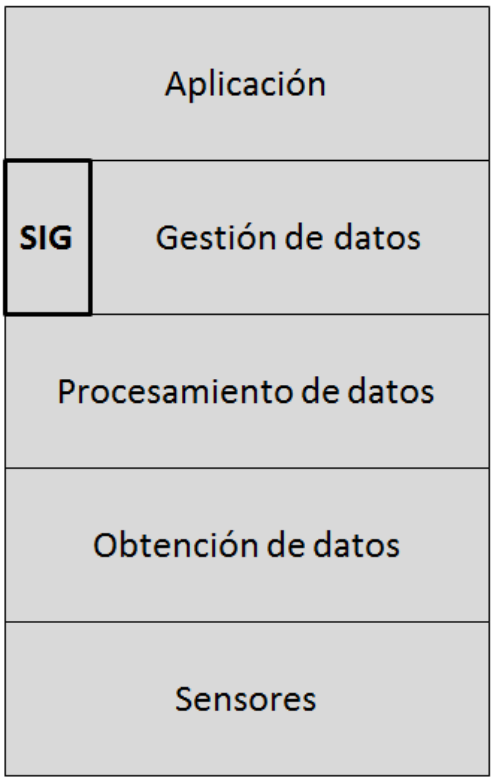


Fig. 24. Módulo SIG incorporado en la arquitectura de la plataforma

3.3.7. RP7 - Colaboración

El concepto de colaboración en el desarrollo y configuración de sistemas sensibles al contexto entre usuarios técnicos y usuarios expertos en el dominio es uno de los pilares fundamentales en el presente trabajo de investigación. Esta colaboración determina dos requerimientos fundamentales de la plataforma, los cuales se describen a continuación.

RP7.1 Entorno colaborativo. El entorno de desarrollo tiene que permitir un trabajo colaborativo entre usuarios técnicos con conocimientos de programación y usuarios no técnicos que tengan un conocimiento exhaustivo del dominio en el que se quiere implantar el sistema sensible al contexto, es decir, expertos en el dominio de aplicación del sistema a implementar que no necesariamente tienen que disponer de conocimientos de programación. La colaboración entre estos dos tipos de usuarios ya ha sido probada en diversos trabajos preliminares (Cassou et al., 2010; Guo et al, 2010; Sohn y Dey, 2003) mostrando unos resultados prometedores.

Esta colaboración puede mejorar el proceso de desarrollo de soluciones que utilicen información sensible al contexto ya que la colaboración da lugar a una menor tasa de errores en el proceso de desarrollo y por consiguiente, el producto final es de mejor calidad (Sohn y Dey, 2003). Además, la implicación en el proceso de desarrollo desde el primer momento por parte usuarios expertos en el dominio de aplicación puede asegurar el cumplimiento de los requerimientos especificados para el desarrollo del sistema final de primera mano (Whitehead, 2007).

RP7.2 Entorno visual. El entorno de desarrollo debe ser visual y no debe estar sujeto a la especificación de código propio de algún lenguaje de programación, ya que este hecho impediría que usuarios sin conocimientos de programación pudieran participar en el proceso.

Además, la interfaz de usuario debe ser lo más usable posible y lo más sencilla de utilizar para ambos perfiles de usuario, pudiendo ofrecer funcionalidades más avanzadas para el perfil programador. Uno de los aspectos a tener en cuenta, es el vocabulario utilizado en la interfaz gráfica de la plataforma, ya que tiene que ser lo más cercano y entendible posible para el usuario no técnico, pero sin alejarse demasiado de la nomenclatura que los usuarios programadores utilizan en el desarrollo de software. Este es un aspecto importante relacionado con la usabilidad que la plataforma tiene que tener para propiciar el trabajo en colaboración. La ISO define la usabilidad como la capacidad que tiene un producto para ser usado por determinados usuarios con el fin de alcanzar unos objetivos concretos con efectividad, eficiencia y satisfacción dentro de un contexto de uso específico (ISO 9241-11). Existen modelos de evaluación de usabilidad, como por ejemplo la "Evaluación Heurística" de Nielsen (Nielsen y Molich, 1990) que incorporan principios heurísticos relacionados con este aspecto, estableciendo que las

interfaces de los usuarios deben utilizar una nomenclatura, términos y conceptos familiares al usuario, en este caso, familiares tanto a usuarios técnicos como no técnicos.

3.3.8. RP8 - Web

La plataforma debe posibilitar su configuración desde cualquier dispositivo y desde cualquier computador conectado a Internet, con el fin de que los comportamientos que se configuren puedan modificarse en tiempo real desde cualquier navegador web, sin necesidad de tener el entorno instalado en el ordenador. Este aspecto está relacionado con los entornos tan dinámicos sobre los que se despliegan las aplicaciones sensibles al contexto, de manera que la plataforma no debe ser un sistema cerrado, sino abierto a cualquier cambio que pueda necesitar el comportamiento del mismo.

Esta necesidad de disponibilidad en línea, se alinea con el entorno visual que debe tener la plataforma, y con uno de los retos y tendencias de la siguiente generación de herramientas y entornos de desarrollo, como es el establecimiento online de las mismas (Whitehead, 2007). Estos nuevos entornos se basarán en interfaces ricas que se asemejen a las herramientas actuales orientadas a soluciones de escritorio, las llamadas Aplicaciones de Internet Enriquecidas o *Rich Internet Applications (RIA)* (Rossi y Sánchez-Figueroa, 2010).

Este requisito se alinea también con el ya mencionado *RP1.3 Computación en la Nube*, por lo que además de exponer servicios a capas de la propia arquitectura, la plataforma desplegada en la nube ofrecería una interfaz web con la que poder gestionar de manera visual la información de contexto.

3.4. Resumen

En este capítulo se ha presentado la base teórica sobre la que se apoyará la plataforma para el desarrollo colaborativo de sistemas sensibles al contexto y la metodología de desarrollo aplicable a la misma descrita en los siguientes capítulos.

Se han definido dos conceptos fundamentales que sirven de pilares principales para el trabajo de investigación realizado, como son los conceptos de contexto y situación. Las dos definiciones planteadas, junto con las carencias y problemas encontrados en la revisión de los entornos de desarrollo de sistemas sensibles al contexto, han dado lugar a una serie de requisitos que deben cumplir tanto la metodología definida como la plataforma para el desarrollo de sistemas sensibles al contexto.

De esta manera, se establece que la metodología debe estar centrada en la definición de las situaciones que el sistema sensible al contexto a desarrollar tiene que ser capaz de detectar. Además, esta especificación y parametrización de situaciones debe realizarse de manera colaborativa entre el experto en el dominio y el programador. El experto en el dominio aportará su conocimiento en el dominio de aplicación en el que se desplegará el sistema a implementar, mientras que el programador contribuirá con sus conocimientos técnicos.

En cuanto a la plataforma de desarrollo, se han especificado diversos requerimientos relacionados con diferentes aspectos de la misma. Por una parte, la arquitectura de la plataforma debe estar influenciada por las buenas prácticas definidas por la comunidad *WoT* para facilitar el acceso a datos provenientes de las fuentes de contexto. Por otra parte, esta arquitectura debe tener en cuenta aspectos relacionados con *EUD*, estableciendo mecanismos para la gestión de los datos de contexto que puedan ser utilizados por usuarios sin conocimientos de programación. Además, la plataforma debe posibilitar su despliegue en infraestructuras en la *nube*, con el fin de externalizar el mantenimiento hardware de los sistemas en los que se despliega y facilitar su adopción por parte de los usuarios de la misma. A su vez, se hace referencia al carácter reactivo y adaptativo que la plataforma debe implementar con el fin de abstraer la gestión de la información de contexto del sistema final a desarrollar.

En cuanto al modelo de datos de contexto que debe ser manejado por los módulos de gestión de la información de contexto de la plataforma, se especifican una serie de requisitos generales, como son la flexibilidad del mismo, el soporte a la aplicación de mecanismos de inferencia y el soporte a la heterogeneidad de datos de contexto y a la definición de dependencias y relaciones entre entidades de contexto. A su vez, el modelo de datos debe posibilitar la representación espacial y temporal, latentes en la propia definición de situación establecida.

Los mecanismos de razonamiento vienen también determinados por una serie de condicionantes. Así, estos mecanismos deben estar basados en reglas con el fin de poder hacer frente al volcado de conocimiento por parte de los expertos en el dominio y a la modificación de los comportamientos definidos en tiempo real, fruto del dinamismo que presenta este tipo de entornos de aplicación. Además, el razonamiento debe posibilitarse a nivel espacial y temporal.

En cuanto a la gestión de la información de contexto que la plataforma tiene que llevar a cabo, es importante que se realice de manera automatizada. Así, la transformación de los datos de contexto al modelo de datos definido, la inserción, actualización y agregación de los datos de contexto debe ser automática con el fin de optimizar la gestión de la información y minimizar los posibles errores. Además, se debe establecer un mecanismo para la recolección de basura, con el fin de eliminar aquella información de contexto que no esté actualizada y optimizar así el rendimiento de la plataforma.

La plataforma debe posibilitar la extensión de los diferentes elementos participantes en la gestión del contexto. De esta manera, se debe facilitar la modificación en tiempo real de las fuentes de contexto, el modelo de datos y las reglas, minimizando el impacto en el resto de la plataforma, además de en el sistema sensible al contexto final a implementar o ya implementado.

Por su parte, la plataforma debe ofrecer mecanismos para gestionar la movilidad de las entidades que así lo requieran, contando con un sistema SIG que sea capaz de realizar operaciones geoespaciales sobre las entidades de contexto que dispongan de datos sobre su ubicación.

El entorno de la plataforma debe facilitar la colaboración entre programadores y expertos en el dominio, proporcionando un entorno visual en el que se puedan desenvolver ambos usuarios con total libertad y teniendo un control total sobre cada uno de los elementos configurables en la misma.

Finalmente, la plataforma debe ofrecer un acceso web con el fin de poder modificar y configurar los diferentes elementos de manera remota a través de cualquier navegador web.

En los dos siguientes capítulos se describen tanto la metodología de desarrollo como la plataforma en base a los requisitos previos identificados en el presente capítulo.

Capítulo 4 *Situation-Driven Development*: metodología de desarrollo colaborativa basada en la definición de situaciones

"Hay que aprender las reglas del juego. Y luego tienes que jugar mejor que nadie"
Albert Einstein (1879 - 1955). Físico alemán.

En este capítulo se describe la metodología de desarrollo diseñada para guiar el proceso de diseño, implementación y configuración de sistemas sensibles al contexto. Esta metodología hace especial énfasis en la colaboración entre expertos en el dominio y programadores.

La metodología, denominada *Situation-Driven Development* o Desarrollo Dirigido por Situaciones, se fundamenta en la identificación y parametrización de cada una de las situaciones referentes a las entidades relevantes para el sistema sensible al contexto a desarrollar. La metodología presenta un proceso colaborativo de identificación y parametrización de dichas situaciones por parte de expertos en el dominio de aplicación y los propios programadores.

En siguientes secciones se describen las bases de la metodología, así como las diferentes fases en las que se descompone la misma. Además, se muestra un caso de uso de ejemplo en el que se ilustra la aplicación de la metodología en el proceso de desarrollo de un sistema sensible al contexto.

4.1. Situation-Driven Development

Como se ha comentado anteriormente, la metodología se fundamenta en la identificación, parametrización y configuración de cada una de las situaciones que el sistema sensible al contexto a desarrollar debe ser capaz de detectar para poder adaptar su comportamiento de manera proactiva en función de las mismas. El ciclo de vida de la metodología se divide en cinco fases diferentes: *análisis*, *configuración*, *desarrollo*, *validación* y *mantenimiento*. A su vez, por cada una de estas fases se encuentran una serie de tareas que deben ser realizadas por parte del programador y el experto en el dominio de manera colaborativa.

El escenario en el que se concibe la metodología es aquel en el que el experto en el dominio puede hacer las veces de cliente del sistema sensible al contexto a implementar y/o de usuario final del propio sistema a desarrollar. De esta manera, es el programador el que dispone de las *herramientas de desarrollo* necesarias para poder involucrar mediante la *metodología* al experto en el dominio en el ciclo de vida del desarrollo del sistema sensible al contexto. Es importante señalar que se toma como premisa que las herramientas de desarrollo utilizadas cumplen con los requisitos expuestos en la Sección 3.3 con el fin de que se pueda materializar la implicación de los expertos en el dominio y la colaboración entre estos y los programadores en las fases de identificación, parametrización y configuración de las situaciones[34].

Un ejemplo concreto de escenario de aplicación de la metodología es aquel en el que la figura del experto en el dominio requiere los servicios de desarrollo de programadores con el fin de implementar un determinado sistema sensible al contexto que posteriormente distribuirá a los usuarios finales. Así, una oficina de turismo que quisiera distribuir una guía turística móvil para proveer de información a los visitantes en movilidad en función del contexto de los mismos, podría requerir servicios externos de programación para desarrollar este sistema sensible al contexto. De esta manera, el personal de la oficina de turismo haría las veces de expertos en el dominio, mientras que los visitantes serían los usuarios finales. Los programadores podrían entonces involucrar a su cliente (oficina de turismo) en el proceso de desarrollo de la guía móvil gracias a la metodología y a las herramientas de desarrollo aplicadas a los sistemas sensibles al

[34] En el Capítulo 5 se muestra la utilización de la metodología de desarrollo sobre la plataforma para el desarrollo de sistemas sensibles al contexto implementada.

contexto. La figura del experto en el dominio puede identificar aquellas situaciones del visitante que tienen que ser detectadas por la guía móvil para poder adaptar la información consultada en función del contexto de alto nivel procesado. Algunos ejemplos de situaciones que podrían ser consideradas por el sistema para adaptar su comportamiento podrían ser los siguientes:

- *Visitante en punto de interés*: cuando un visitante se encuentra cerca de un punto de interés, el dispositivo móvil le muestra de manera automática información sobre el mismo.
- *Visitante esperando al autobús*: cuando un visitante se encuentra esperando al autobús, la guía le informa sobre los horarios de llegada o los puntos de interés a los que puede llegar mediante transporte público desde el punto en el que se encuentra.
- *Visitante de ruta por la ciudad en un día soleado*: cuando un visitante se encuentra de ruta por la ciudad y hace un día de sol y calor, la guía recomienda al visitante actividades turísticas al aire libre.

Así, cada una de las situaciones anteriores podrían ser parametrizadas y configuradas de manera colaborativa entre el programador y el experto en el dominio, en función de las fases establecidas en la metodología y por mediación de la herramienta de desarrollo.

La Figura 25 muestra cada una de las cinco fases que componen la metodología así como las diferentes tareas que tienen que ser llevadas a cabo en cada una de ellas. También se indican los actores (experto en el dominio, programador y usuario final) que pueden participar en cada una de las fases y tareas. No todas las tareas están pensadas para poder ser realizadas en colaboración entre el experto en el dominio y el programador. Existen tareas establecidas en las que solamente puede participar el programador ya que requieren de algún tipo de conocimiento técnico o capacidad de programación.

En cuanto al ciclo de vida de las diferentes fases, se establece que estas sean realizadas de una manera iterativa e incremental, es decir, que por cada una de las situaciones identificadas, se sigan todos los pasos de la metodología hasta la última fase de validación. De esta manera, el sistema sensible al contexto se desarrollará a medida

que se vayan incorporando en el funcionamiento del mismo la detección de las situaciones identificadas.

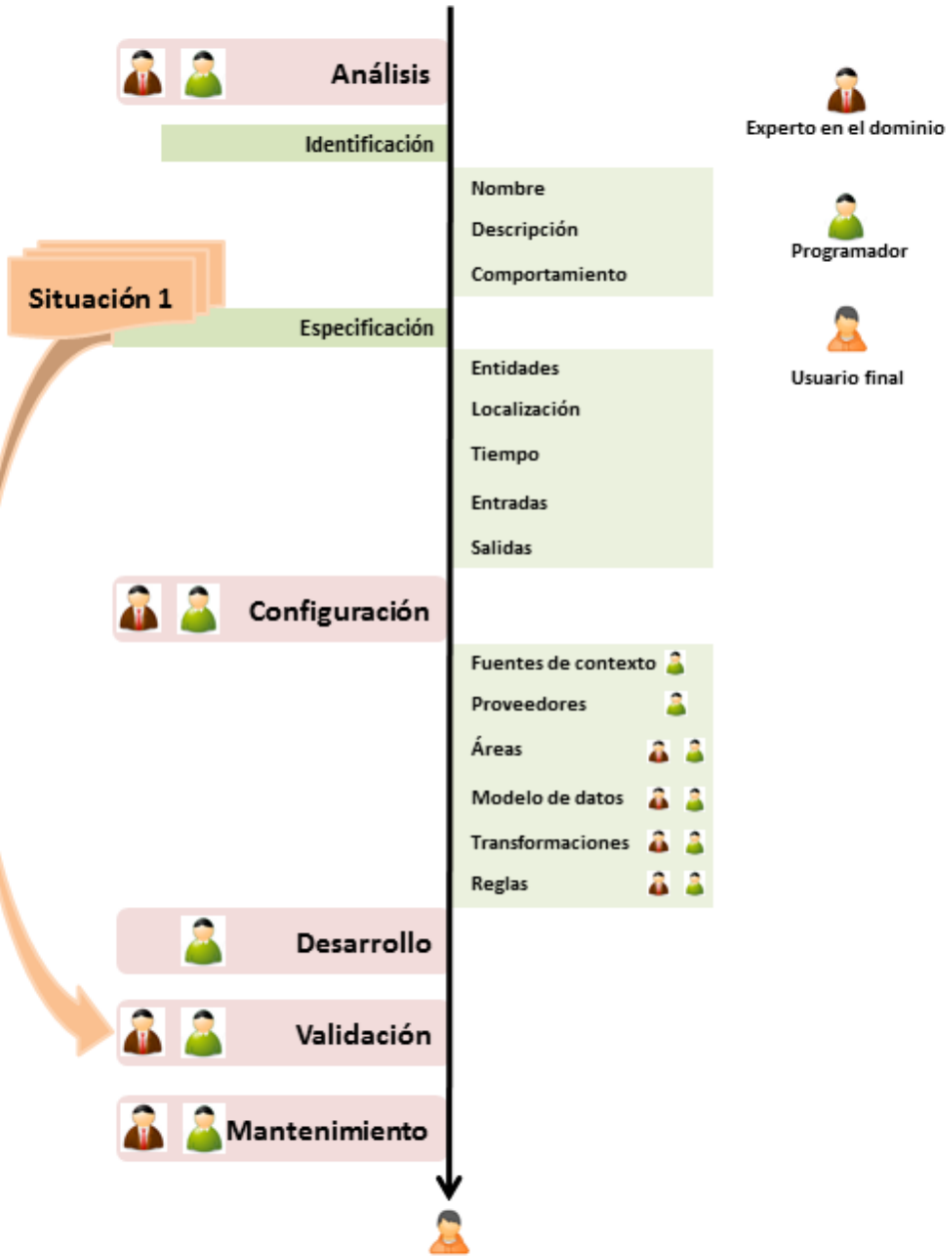


Fig. 25. *Situation-driven Development*

Esta manera incremental de tratar las situaciones identificadas se fundamenta en metodologías ágiles como *Scrum* (Scharff y Verma, 2010), donde se plantean ciclos de trabajo de corta duración (pocas semanas) que aportan valor a la solución final de manera incremental. Cada ciclo de trabajo se denomina *sprint* y se asemeja a los ciclos de trabajo propuestos en la metodología para tratar cada una de las situaciones identificadas. Estos ciclos de trabajo tienen objetivos concretos y posibilitan que se

valide la solución a medida que se tratan las situaciones identificadas, con lo que se puede responder de manera más inmediata a posibles cambios en los requerimientos iniciales, reduciendo además el riesgo de implementar funcionalidades equivocadas.

En siguientes secciones se describe en detalle cada una de estas fases así como las funciones de los diferentes actores en cada una de ellas.

4.1.1. Análisis

En la fase de análisis se deben identificar todas aquellas situaciones que resultan relevantes para el funcionamiento del sistema. En esta fase participan tanto el programador como el experto en el dominio, teniendo este último más peso en el proceso, ya que es la figura que mejor conoce las situaciones que tienen que ser identificadas por el sistema a implementar. El programador actúa en este caso como consultor tecnológico, sirviendo de apoyo en la definición de las situaciones.

Esta primera fase se descompone a su vez en dos tareas fundamentales, como son la identificación de las situaciones y la especificación general de las mismas. A continuación se describen ambas tareas.

4.1.1.1. **Identificación de situaciones**

En la fase de identificación, se deben listar todas aquellas situaciones relevantes para el sistema, proporcionando además una descripción general de las mismas. Esta fase es fundamental y responde a la captura de requisitos del sistema sensible al contexto a implementar. Es aquí donde el programador y el experto en el dominio tienen que establecer un diálogo fluido para poder identificar en primera instancia las situaciones relevantes que posteriormente serán especificadas, configuradas y utilizadas por el sistema sensible al contexto para adaptar su comportamiento.

En este proceso, por cada situación identificada se deben especificar los siguientes parámetros.

- *Nombre*: nombre identificativo de la situación. Se establecen dos aproximaciones para poder asignar el nombre de una situación.
    - *Lógica*: un nombre lógico puede no tener un significado claro a simple vista, sino que puede ser cualquier representación alfanumérica que sirva para identificar la situación. Los nombres "situación 1", "S2", "s342a"son ejemplos de nombres lógicos.

  - *Descriptiva*: un nombre descriptivo es aquel que da más detalles significativos sobre la situación que identifica. Por ejemplo, "visitante alemán paseando por la ciudad" y "andando por la playa en un día soleado" son nombres descriptivos.

- *Descripción*: se debe establecer una descripción general de la situación especificando el máximo número de detalles posibles. Esta descripción debe contener información sobre cuándo y cómo puede llegar a identificarse la situación. Esta descripción servirá como referencia para posteriores fases donde se tiene que concretar a un nivel de detalle mayor todos los parámetros que componen la situación y participan en su detección.

- *Comportamiento*: se tiene que describir el comportamiento del sistema una vez se identifique la situación. Esta descripción puede ser informal. De esta manera, y como se ha comentado anteriormente en la Sección 3.3.1, el sistema a implementar tendrá un carácter reactivo en función de la situación detectada. La especificación por parte del experto en el dominio del comportamiento deseado en el sistema a implementar dará al programador unas primeras pautas para poder empezar a diseñar el sistema sensible al contexto que el usuario final utilizará.

4.1.1.2. **Especificación de situaciones**

Una vez se hayan identificado todas las situaciones que el sistema debe ser capaz de detectar, se deben especificar también los parámetros que se tendrán en cuenta a la hora de posibilitar la detección de las mismas por parte de la plataforma, herramienta o entorno de desarrollo con el que se cuente para ayudar en la implementación del sistema sensible al contexto. Esta especificación se basa en los conceptos establecidos por Oh et al. (2006), denominados 4W1H, descritos en la Sección 3.2.1.

Así, por cada una de las situaciones identificadas, se deben especificar los siguientes parámetros.

- *Entidades*: como entidades se entienden aquellos seres vivos, objetos o lugares que pueden encontrarse en la situación a definir. Además, también forman parte del grupo de entidades aquellas que participan de una manera indirecta en la propia situación. En otras palabras, la identificación de entidades da respuesta a quién o qué puede encontrarse en la situación que

se está detallando y quién o qué participa en la misma. Como ejemplo, se puede considerar la situación "de turismo en un día soleado", donde se podría identificar que participan en dicha situación las entidades "Visitante" y "Ciudad". La entidad "Visitante" es la que se encuentra en la situación identificada y la entidad "Ciudad" es aquella que participa en la misma, teniendo en cuenta más concretamente, la temperatura ambiental relativa a la propia ciudad. Esta primera identificación de entidades ayudará posteriormente a diseñar el modelo de datos que soporte la información de contexto necesaria en el sistema.

- *Localización*: este parámetro hace referencia al lugar o el área geográfica donde puede ser identificada la situación. En concreto, se propone especificar esta localización utilizando una doble granularidad.
    - *Tipo de área*: mediante esta granularidad de carácter general se debe especificar el tipo de espacio geográfico al que pertenece el área donde se puede detectar la situación. Ejemplos de tipo de área pueden ser los siguientes: playa, ciudad, plaza.
    - *Área*: esta especificación permite determinar la región del espacio específica, perteneciente a un tipo de área concreto, donde se puede detectar la situación. Ejemplos de áreas pueden ser los siguientes: Playa de Gros (playa), Irún (ciudad), Plaza San Juan (plaza).
- *Tiempo*: mediante este parámetro se especifica el rango de fecha y hora en el que la situación puede tener lugar. Por lo tanto, este parámetro hace referencia a la temporalidad en la que una situación se determina. Aun así, puede darse el caso de que el rango temporal no sea determinante a la hora de detectar una situación.
- *Entradas*: una vez que se han definido todos los parámetros anteriores, el siguiente paso es la propia especificación de los datos de contexto o entradas necesarias para que la plataforma o entorno de desarrollo del que se dispone sea capaz de identificar las situaciones descritas y pueda generar las salidas pertinentes para que el sistema final pueda adaptar su comportamiento, tal y como se describe en el patrón de gestión del contexto

propuesto en la Sección 3.3.1. De esta manera, las entradas deben ser descritas utilizando los siguientes parámetros.

- *Dato*: nombre del dato de contexto necesario. Existen multitud de tipos de datos de contexto que pueden ser utilizados con el fin de detectar la situación objetivo (Sección 2.2).
- *Objetivo*: se debe describir de manera detallada el objetivo para el que se utiliza el dato identificado. Por ejemplo, la temperatura de una habitación puede ser utilizada para determinar si hace calor o no en la misma.
- *Condición*: condiciones que debe cumplir el dato. Estas condiciones están ligadas a la propia definición de la situación. Así, se pueden determinar multitud de condiciones que debe cumplir el dato requerido. Siguiendo con el ejemplo anterior de la temperatura de una habitación, se pueden considerar condiciones como las que establecen que dicha temperatura tiene que ser mayor, menor o igual que una determinada medición.
- *Restricciones*: restricciones del dato. Un dato puede tener ciertas restricciones de formato, unidades, etc.
- *Fuente*: aquí se ha de especificar la fuente de contexto de la que se puede obtener el dato. Como ya se ha comentado en la Sección 2.5, estas fuentes de contexto pueden ser lógicas, físicas o virtuales.
- *Comentarios*: comentarios respecto al dato de contexto que pueden ser relevantes de cara a la implementación del sistema.

- *Salidas*: una vez identificadas las entradas, se deben especificar los datos que deben ser producidos por el entorno o plataforma utilizada para la identificación de situaciones, como consecuencia de la gestión e interpretación de los datos de contexto obtenidos y que el programador utilizará para implementar el comportamiento definido en el sistema final. Es en esta definición de los datos de salida donde el programador tiene un mayor peso, ya que son estos datos los que servirán para poder adaptar el funcionamiento del sistema a implementar. El experto en el dominio puede dar su visión teniendo en cuenta su conocimiento del dominio y los datos que pudieran ser relevantes para el propio funcionamiento del sistema a

implementar. Las salidas tienen que ser definidas utilizando los siguientes parámetros.

- *Dato*: nombre del dato de salida. Estos datos pueden ser algunos de los datos utilizados como entrada para la detección de la situación, o datos de alto nivel inferidos a partir de los mismos.
- *Objetivo*: se debe especificar para qué se utilizará el dato de salida en el sistema a desarrollar.
- *Condición*: al igual que las entradas, el dato de salida puede tener algún tipo de condición que deba cumplir.
- *Restricciones*: se deben especificar aquellas restricciones que pueda tener el dato.
- *Comentarios*: comentarios adicionales al dato de salida.

4.1.1.3. **Ficha de recogida de datos**

Para simplificar la identificación, especificación y recogida de los datos que se precisan en esta primera fase de análisis, se ha diseñado la ficha que se muestra a continuación, la cual tiene que ser completada de manera colaborativa por el experto en el dominio y el programador.

Tabla 3. Ficha de recogida de datos para la fase de Análisis

Nombre						
Descripción						
Comportamiento						
¿Quién? ¿Qué? Entidades						
¿Dónde? Tipo Área / Área						
¿Cuándo? Rango de fecha y hora						
ENTRADAS						
#	**Dato**	**Objetivo**	**Condición**	**Restricciones**	**Fuente**	**Comentarios**
SALIDAS						
#	**Dato**	**Objetivo**	**Condición**	**Restricciones**	**Comentarios**	

4.1.2. Configuración

Después de la especificación de los parámetros necesarios para identificar la situación en curso, hay que proceder a la configuración de la plataforma o entorno de desarrollo detallando en la misma todos los parámetros identificados. Esta configuración se realiza en base a las diferentes capas propuestas para la arquitectura de los sistemas sensibles al contexto (Sección 3.3.1) y al marco teórico establecido (Sección 3.1) junto con los requerimientos planteados para la plataforma de desarrollo (Sección 3.3).

Así, esta fase de configuración se divide en diferentes etapas. Algunas configuraciones solamente pueden ser llevadas a cabo por los programadores (P), ya que requieren conocimientos técnicos. El resto de configuraciones pueden ser llevadas a cabo por el experto en el dominio en colaboración con el programador.

- *Fuentes de contexto (P)*: se tienen que configurar las fuentes de contexto identificadas para que la información sea accesible o pueda ser enviada a la plataforma de desarrollo utilizada para la detección de situaciones. Esta configuración puede requerir algún tipo de desarrollo previo por parte de los programadores con el fin de adaptar los sensores o fuentes de datos de contexto utilizados. Por ejemplo, un sensor puede requerir una implementación de capas superiores que sean accesibles por la plataforma de desarrollo con la que se cuente con el fin de que se comunique con los controladores de dicho sensor de una manera transparente para poder obtener los datos de contexto.

- *Proveedores (P)*: los proveedores son los componentes software que se establecen en la capa de obtención de datos con el fin de capturar los datos de contexto necesarios por la plataforma. Como se ha visto en la Sección 2.8., esta capa de proveedores es utilizada por la mayoría de los entornos de desarrollo como pasarela de comunicación entre la capa de sensores y la capa de gestión de la información de contexto. Así, se encuentran módulos software como los *widget* del entorno *Context Toolkit* (Dey et al., 2001) o los *wrapper* del entorno *Semantic Space Toolkit* (Wang et al., 2004). Por lo tanto, esta configuración de los proveedores puede requerir de algún tipo de conocimiento técnico para que la plataforma pueda obtener los datos de las diferentes fuentes identificadas.

- *Áreas*: en base al marco teórico establecido, las situaciones se localizan en áreas determinadas, por lo que es necesaria la configuración tanto de las áreas como de los tipos de áreas especificados en la fase de análisis. Esta configuración de las áreas tiene que ser posibilitada por el módulo SIG que la plataforma debe exponer a los usuarios de la misma, tal y como se recoge en la Sección 3.3.6. Esta configuración tiene que poder ser realizada tanto por los expertos en el dominio como por los propios programadores.

- *Modelo de datos*: tras la definición de las regiones geográficas en las que se puede identificar la situación, es necesaria la creación del modelo de datos que sirva para representar la información de contexto proveniente de las fuentes de contexto identificadas. Esta creación del modelo de datos debe basarse en las entidades participantes en la situación, especificadas anteriormente en la fase de análisis. De esta manera, se posibilita la agregación de datos de contexto provenientes de diferentes fuentes de contexto heterogéneas, facilitando su gestión por parte de la plataforma. El modelo de datos tiene que ser definido también tanto por los expertos en el dominio como por los programadores de manera colaborativa, por lo que se deben ofrecer entornos de configuración simples, tal y como se especifica en la Sección 3.3.7, en relación al requisito *RP7.2 Entorno visual.*

- *Transformaciones*: en esta fase de configuración de la plataforma se tienen que poder realizar las transformaciones pertinentes para posibilitar que los datos provenientes de las fuentes de contexto puedan ser representados en el modelo de datos definido anteriormente. Esta configuración hace referencia a la capa de procesamiento de datos propuesta en la arquitectura definida en la Sección 3.3.1. Además, tal y como se menciona en el requisito *RP1.2 Desarrollo Orientado al Usuario Final*, este tipo de transformaciones tienen que poder realizarse tanto por los programadores como por los expertos en el dominio.

- *Reglas*: finalmente y en base a la descripción y especificación de la situación y de los datos de entrada definidos junto con sus condiciones y restricciones, se deben configurar las reglas lógicas que sirvan para detectar dicha situación. Estas reglas responden al mecanismo de identificación de

situaciones propuesto para el presente trabajo de investigación y descrito en los requerimientos especificados en la Sección 3.3.3. Estas reglas tienen que poder ser definidas teniendo en cuenta los parámetros temporales y espaciales definidos en la fase de análisis y tienen que posibilitar además su creación de manera colaborativa por parte de expertos en el dominio y los programadores. Además, las reglas definidas en la plataforma o herramienta de desarrollo tienen que permitir la generación de los datos de salida especificados en la fase de análisis para que sean utilizados en la siguiente fase por el programador, donde implementará los comportamientos definidos en base a dichas salidas y situaciones identificadas.

4.1.3. Desarrollo

La fase de desarrollo comprende la implementación por parte del programador del sistema sensible al contexto que utilizarán los usuarios finales. Este sistema será el que adapte su comportamiento en función de las salidas generadas por la plataforma de desarrollo al identificar cada una de las situaciones especificadas en la fase de análisis y configuradas posteriormente para que la plataforma las detecte. Esta fase de desarrollo puede iniciarse con independencia de que las situaciones hayan sido configuradas, pero deberá acoplarse al ciclo de vida de la metodología en cuanto las funcionalidades de la misma requieran algún tipo de información de contexto y se tengan que utilizar las salidas generadas al detectar las situaciones relevantes para el sistema.

Es interesante, como ya se ha comentado anteriormente, que por cada situación analizada y configurada, se desarrolle la funcionalidad del sistema que requiere dicha situación para adaptar el comportamiento del mismo, de tal manera que el sistema evolucione de manera incremental. De esta manera, se podrán tener las impresiones del propio cliente (experto en el dominio) de primera mano y se podrán corregir errores o reaccionar a cambios en los propios requerimientos de la solución a implementar de manera más ágil. Para verificar que todo el proceso se realiza de manera satisfactoria, se encuentra la siguiente fase, la fase de validación.

4.1.4. Validación

Una vez se haya implementado el comportamiento del sistema a desarrollar en base a la situación en curso identificada, tanto los programadores como los expertos en el dominio tienen que validar que el sistema cumpla con las especificaciones requeridas en

la fase de análisis de la situación en curso. Si es necesario, se volverán a repetir las fases de configuración y desarrollo para poder adaptar los comportamientos del sistema a las especificaciones iniciales de la situación. Una vez se haya validado el ciclo completo de una situación, se procederá a realizar la misma operación con el resto de situaciones a identificar si las hubiere.

4.1.5. Mantenimiento

La última fase es la relacionada con el mantenimiento del sistema sensible al contexto implementado, estrechamente relacionada con la gestión de la información de contexto llevada a cabo por la plataforma o herramienta disponible. Esta fase se da una vez que se ha desarrollado la solución por completo y consiste en mantener la misma bajo unos estándares determinados de calidad y de funcionalidad. Es interesante que en esta fase de mantenimiento, los propios expertos en el dominio puedan ser capaces de realizar cambios en la gestión de los datos de contexto afectando lo menos posible al sistema implementado (Trigg et al., 1987). Este mantenimiento en la gestión de los datos de contexto puede implicar, por ejemplo, la creación de nuevas áreas o tipos de áreas donde se pueda detectar una determinada situación anteriormente creada, la modificación de las reglas definidas por cambios en los requerimientos que afectan a las condiciones de los datos de contexto o incluso, pueden requerirse nuevas fuentes de contexto para detectar de una manera más precisa las situaciones configuradas. Tal y como se recoge en la Sección 3.3.5 y más concretamente en el requisito de extensibilidad propuesto, la plataforma para la gestión de los datos de contexto tiene que permitir este tipo de cambios en tiempo de ejecución, afectando lo menos posible al sistema sensible al contexto desplegado.

4.2. Caso de uso de ejemplo

En esta sección se describe un breve ejemplo que contiene los detalles de la fase de análisis de una situación. La situación planteada servirá además como base del Capítulo 5, donde se describen las configuraciones a realizar para poder detectar dicha situación mediante la plataforma para el desarrollo de sistemas sensibles al contexto implementada.

Además, se muestra la arquitectura general del sistema sensible al contexto resultante que hará uso de la situación, donde figuran tanto la plataforma de desarrollo como la guía turística móvil a implementar.

4.2.1. Análisis de la situación

A continuación se describe un ejemplo o caso de uso donde se especifica una situación en la que se puede encontrar un usuario final en un dominio turístico. En este escenario, el usuario final (visitante) dispone de una guía móvil que hace uso de información de contexto para personalizar la información que se muestra. Se ha identificado previamente una situación por parte del experto en el dominio turístico (oficina de turismo) que resulta relevante para que el servicio móvil pueda reaccionar a la misma.

Así, la situación de ejemplo es la siguiente: *un visitante de origen alemán está realizando una ruta por la ciudad de Donostia-San Sebastián en un día soleado y a la hora de comer*. Detectando esta situación, un sistema sensible al contexto podría, por ejemplo, enviar al dispositivo móvil del turista un cupón de descuento para ir a comer a la terraza de un restaurante cercano a su posición.

En la siguiente tabla se muestra la ficha con los parámetros relativos a la fase de identificación de la situación de ejemplo.

Tabla 4. Identificación de la situación "De ruta alemán - sol - hora de comer"

Nombre	De ruta alemán - sol - hora de comer
Descripción	Un visitante de origen alemán se encuentra realizando una ruta por la ciudad de Donostia-San Sebastián en un día soleado y es la hora de comer.
Comportamiento	El usuario recibe en el dispositivo móvil un cupón de descuento para ir a comer a la terraza de un restaurante cercano a su posición.

En las tablas que siguen a continuación se muestra la especificación realizada sobre la situación identificada anteriormente. En la Tabla 5 se especifican los datos relativos a las entidades que participan en la situación, la localización de la misma y el rango temporal.

Tabla 5. Especificación de la situación "De ruta alemán - sol - hora de comer" (a)

¿Quién? Entidades	Persona (visitante)
¿Dónde? Tipo Área /Área	Ciudad / Donostia-San Sebastián
¿Cuándo? Rango de fecha y hora	Cualquier día de la semana. Hora de comer, entre las 12:00 y 13:00 (por ser de origen alemán).

Como puede observarse, la entidad que participa en la situación es la propia persona o visitante como tal. La localización en la que se da la situación es, en este caso

concreto, la ciudad de Donostia-San Sebastián, teniendo como rango temporal cualquier día de la semana entre las 12:00 y las 13:00, hora aproximada para comer en Alemania.

Además, en la Tabla 6 se identifican los datos de entrada a la plataforma necesarios para detectar tal situación (datos de contexto de bajo nivel), así como las salidas que se producen por parte de la plataforma (situaciones) y que serán utilizadas por el sistema que quiera adaptarse a las mismas, en este caso, por el servicio de envío de cupones de descuento a las guías móviles de los visitantes.

Tabla 6. Especificación de la situación "De ruta alemán - sol - hora de comer" (b)

ENTRADAS						
#	**Dato**	**Objetivo**	**Condición**	**Restricciones**	**Fuente**	**Comentarios**
1	Localización de la persona (latitud, longitud)	Detectar que una persona está en Donostia-San Sebastián	Localización = área Donostia-San Sebastián		GPS (dispositivo móvil)	
2	Temperatura de la ciudad	Conocer si hace calor en la ciudad	Temperatura >= 25ºC	Unidades Celsius	Servicio web meteorológico	
3	País de procedencia de la persona	Identificar a las personas de origen alemán	Origen = Alemania		Guía móvil / perfil de usuario (dispositivo móvil)	
4	Identificador de la persona	Identificar la persona que se encuentra en la situación definida			Guía móvil / perfil de usuario (dispositivo móvil)	Número de teléfono utilizado en el registro de la guía móvil.
SALIDAS						
#	**Dato**	**Objetivo**	**Condición**	**Restricciones**	**Comentarios**	
1	Nombre de la situación	Identificar la situación requerida				
2	Identificador de la persona	Enviar el cupón al usuario concreto			Número de teléfono utilizado en el registro de la guía móvil.	
3	Localización de la persona (latitud, longitud)	Recomendar un restaurante cercano				

4.2.2. Arquitectura general del sistema

La Figura 26 muestra la arquitectura general del sistema a implementar. En esta arquitectura se refleja tanto la plataforma de desarrollo como el propio sistema sensible al contexto a ser desarrollado. Este último tiene como componentes principales un servicio de mensajería y la guía móvil.

La guía móvil puede estar desarrollada para cualquier sistema operativo móvil existente (*Android*, *iOS*, etc.) ya que la plataforma es independiente del lenguaje de programación en el que se implemente el sistema final, en este caso, la propia guía. En la figura se han representado tres usuarios de la guía (A, B y C), por lo que la plataforma es capaz de gestionar toda la información de contexto que llega de cada uno de los usuarios de la guía móvil. En este caso concreto, y teniendo como referencia el análisis realizado de la situación de ejemplo, actuarán como parámetros de entrada a la plataforma la localización del usuario, el origen, el identificador del mismo y la temperatura de la ciudad.

El servicio de mensajería utilizará las salidas de la plataforma para poder enviar los cupones de descuento adecuados (referentes al restaurante más cercano) a las guías instaladas en los dispositivos móviles. Estas salidas contendrán el nombre de la situación detectada, el identificador del usuario, que en este caso será el teléfono móvil, y su localización.

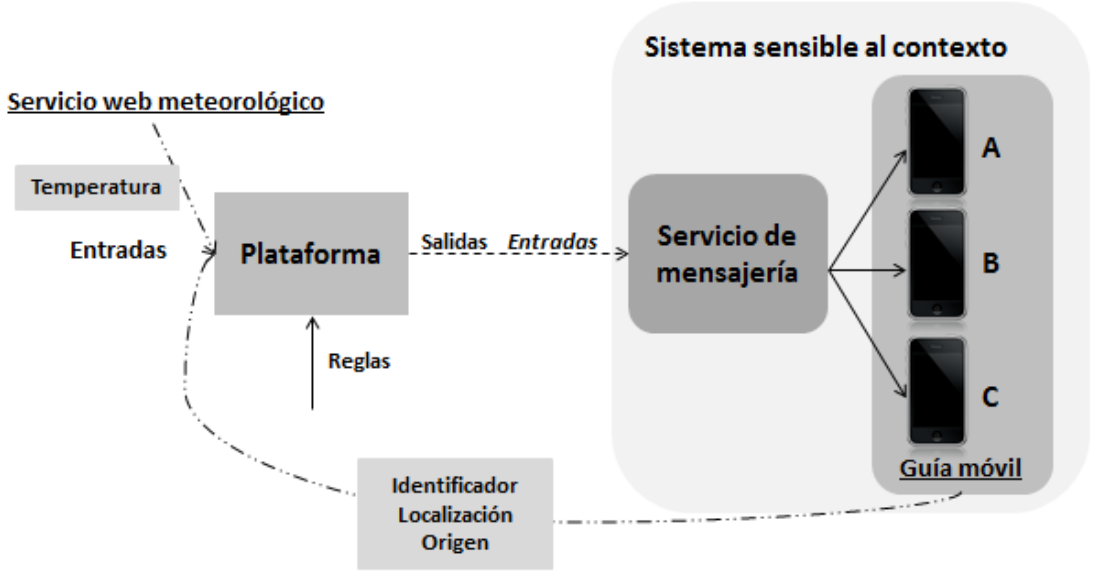


Fig. 26. Arquitectura del sistema sensible al contexto de ejemplo

Un servicio implementado e incorporado en la propia guía se encargaría de enviar a la plataforma la información de contexto relativa al visitante para que pudiera ser gestionada por la misma. Así, el identificador y el origen del visitante podrían ser enviados al configurar el perfil de usuario en la guía móvil, mientras que la localización tendría que ser enviada periódicamente con el fin de gestionar adecuadamente esta información de contexto. El visitante podría descargarse la guía móvil en la oficina de turismo de la ciudad para poder disfrutar de información turística personalizada en función de su contexto.

En el Capítulo 5 se ilustra la configuración de la plataforma para el desarrollo del sistema sensible al contexto descrito, en base a la metodología propuesta y las especificaciones para la situación de ejemplo tratada.

4.3. Resumen

En esta sección se ha descrito la metodología de desarrollo diseñada para identificar y especificar las diferentes situaciones que resulten relevantes para los sistemas sensibles al contexto a desarrollar. Esta metodología se divide en diferentes etapas que han de ser seguidas de una manera iterativa e incremental por los programadores en colaboración con los expertos en el dominio.

Se plantean cinco fases: análisis, configuración, desarrollo, validación y mantenimiento. Estas fases se basan tanto en el marco teórico establecido en el presente trabajo de investigación, en relación a los requerimientos de la plataforma para apoyar el desarrollo de sistemas sensibles al contexto, como los propios requerimientos de la metodología de desarrollo. De esta manera, la metodología tiene como premisa que el equipo de programación dispone de una plataforma de apoyo al desarrollo de sistemas sensibles al contexto, enfocada a la gestión de los datos de contexto necesarios.

Así, la metodología propuesta cumple con los requisitos identificados en la definición del marco teórico.

- *RM1 - Centrada en la definición de situaciones*. La metodología presentada está guiada por la identificación de todas las situaciones relevantes para el sistema y la especificación de todos los parámetros necesarios para que la plataforma pueda identificarlas. Estos parámetros se plasman a posteriori en la fase de configuración sobre la plataforma de desarrollo, con el fin de que las situaciones puedan ser identificadas.

- *RM2 - Colaborativa*. Al contrario que las metodologías estudiadas (Green y DiCaterino, 1998; Henricksen e Indulska, 2006; Hirschfeld et al., 2008), esta metodología se centra no solamente en la figura del programador, sino también en el experto en el dominio de aplicación que puede ser involucrado en etapas tempranas del desarrollo. Sí que es cierto que algunas fases tienen que ser llevadas a cabo por el programador exclusivamente, debido a su carácter tecnológico, pero estas son las menos, posibilitando así que ambos puedan realizar un trabajo colaborativo desde un primer momento. Este trabajo en colaboración desde las fases de análisis iniciales posibilita que la solución final se adapte mejor a los requerimientos planteados inicialmente por el cliente, que en este caso es la figura del experto en el dominio. Además, permite reaccionar de una mejor manera a los posibles cambios en los requerimientos planteados en las fases iniciales. Esto convierte a esta metodología en una metodología ágil, donde se permite dar respuesta a cambios en los requerimientos de una manera más inmediata, involucrando además al cliente desde las fases iniciales del mismo (Kent Bleck, 2004).

Capítulo 5 *Context Cloud*: plataforma para el desarrollo colaborativo de sistemas sensibles al contexto

"La función de un buen software es hacer que lo complejo aparente ser simple"

Grady Booch (1955). Ingeniero de software americano.

En base a la hipótesis y objetivos planteados en el presente trabajo de investigación, las carencias encontradas en el estado del arte, el marco teórico definido, los requerimientos identificados y la metodología de desarrollo propuesta, se ha diseñado e implementado la plataforma *Context Cloud*[35]. El objetivo principal es **dar soporte al desarrollo colaborativo de sistemas sensibles al contexto** siguiendo la metodología descrita en el Capítulo 4, teniendo a su vez dos objetivos más concretos.

- Simplificar el desarrollo de sistemas sensibles al contexto ofreciendo soporte y funcionalidades propias para facilitar este tipo de desarrollos.
- Ofrecer un entorno de desarrollo en el que gente técnica con conocimientos de programación y gente no técnica, como los expertos en el dominio de aplicación, puedan colaborar en el proceso de desarrollo, mejorando así tanto el propio proceso de implementación como el producto o servicio final.

[35] El nombre de la plataforma, *Context Cloud* o Nube de Contexto, se basa en la metáfora dada por la acumulación de información de contexto en la infraestructura en nube sobre la que se despliegan los diferentes componentes de la plataforma.

La plataforma se basa en la obtención de información de contexto de diferentes fuentes (móviles, páginas web, sensores, etc.) con el fin de gestionarla e interpretarla en beneficio de los sistemas que se quieran adaptar a dichos datos de contexto. Para ello, la plataforma se centra en la detección de las situaciones en las que se encuentran las entidades relevantes (ser vivo, objeto o lugar) para el sistema a implementar. Estas situaciones se consideran como una abstracción de alto nivel sobre el conjunto de información de contexto utilizado para la detección de las mismas.

Para que un sistema pueda identificar situaciones, se necesitan multitud de datos de contexto, como por ejemplo, la temperatura ambiente, la localización de la entidad, su perfil, la manera en la que se mueve, la fecha y hora del día, etc. La agregación, composición e interpretación de todos estos datos es la que posibilita el reconocimiento de situaciones que ayudan a personalizar el comportamiento de los sistemas, convirtiéndose así en sistemas sensibles al contexto.

Como se ha comentado con anterioridad, una persona experta en el dominio de aplicación, es capaz de definir de mejor manera las situaciones en las que se puede encontrar la entidad relevante para el sistema. Así, mientras que el programador es el responsable de implementar el sistema sensible al contexto en la fase de desarrollo planteada en la metodología (Sección 4.1.3), el experto en el dominio puede colaborar en la definición y configuración de los comportamientos o situaciones junto con el propio programador en las demás fases que componen la metodología.

Así, *Context Cloud* permite realizar de manera colaborativa entre el programador y el experto en el dominio, la gestión de los datos de contexto requeridos para identificar las diferentes situaciones y proporcionar esta información a los sistemas que quieran adaptarse a dichas situaciones. La gestión de la información de contexto incluye las siguientes funcionalidades principales basadas en la arquitectura propuesta en la Sección 3.3.1.

- Obtención de datos de las diferentes fuentes de contexto identificadas.
- Agregación y composición de los datos de contexto obtenidos.
- Transformación de los datos de contexto a un modelo de datos procesable por la plataforma.

- Interpretación e identificación de situaciones de alto nivel en base al modelo de datos de contexto definido y reglas lógicas.

De esta manera, la plataforma *Context Cloud* puede considerarse como una *caja negra* en la que entran datos provenientes de las diferentes fuentes de contexto identificadas y se procesan en base a las reglas de comportamiento que la plataforma permite definir. Estas reglas desencadenan salidas con información de contexto de alto nivel referente a las situaciones identificadas que servirán a su vez como entrada a los servicios que se quieran contextualizar, tal y como se establece en el patrón de computación de información de contexto descrito en la Sección 3.3.1. En la figura a continuación se muestra un esquema de este flujo de interacción, junto con los usuarios fundamentales de la plataforma.

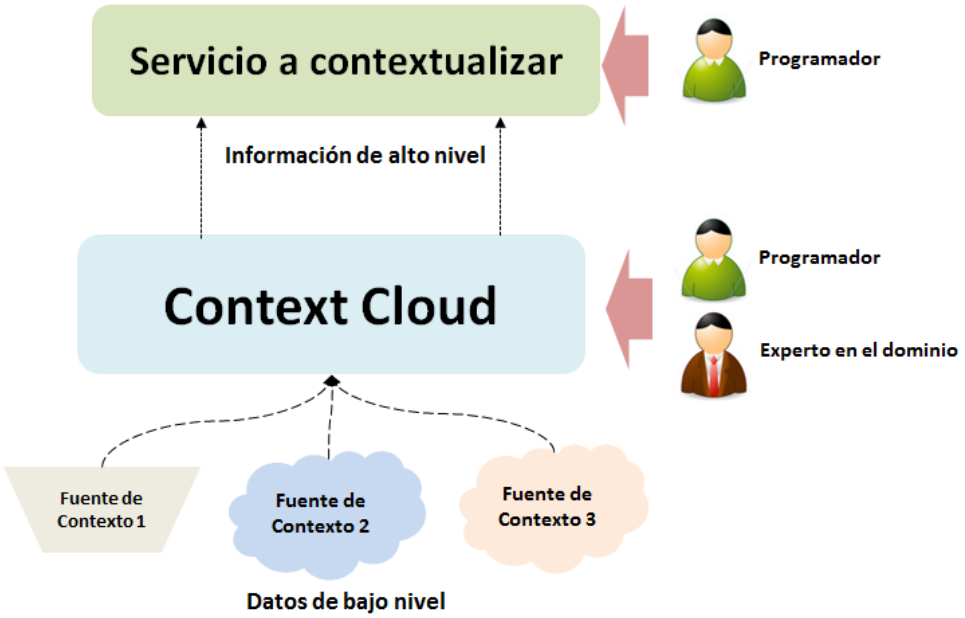


Fig. 27. Flujo de interacción con la plataforma

Como se puede observar en la figura, se han ilustrado tres fuentes de datos de contexto diferentes, de las que la plataforma obtiene datos de bajo nivel, es decir, datos que no han sido procesados ni agregados. *Context Cloud*, ofrece una interfaz web de usuario mediante la cual, tanto programadores como expertos en el dominio pueden configurar en colaboración los diferentes parámetros necesarios para la obtención, gestión y procesamiento de los datos de contexto, todo ello basándose en la fase de análisis de la metodología de desarrollo propuesta (Sección 4.1.1). Fruto de este

procesamiento, la plataforma es capaz de generar salidas de datos de alto nivel que pueden servir como entrada a los servicios o sistemas a contextualizar.

La plataforma permite la identificación de las situaciones de las entidades relevantes para el sistema a personalizar mediante la definición de reglas lógicas aplicadas a la información de contexto obtenida. Además, estas situaciones se asocian a zonas geográficas concretas, las cuales pueden ser también definidas mediante la interfaz web de la plataforma. Es decir, que para un área determinada, la plataforma permite la configuración de diferentes reglas en base a diferentes parámetros y valores de contexto definidos en el modelo de datos, que sirven para identificar las situaciones en las que se puede encontrar una entidad determinada cuando se encuentra en dicha área.

En las siguientes secciones se describen tanto la arquitectura de la plataforma, como cada una de las funcionalidades ofrecidas para poder configurar la misma en base a las diferentes especificaciones realizadas en la fase de análisis de la metodología.

Para ilustrar el funcionamiento de la misma, se va a continuar con el ejemplo expuesto en el capítulo anterior referente a la metodología de desarrollo y en el que se recoge un análisis completo de la siguiente situación.

"Visitante de origen alemán de ruta por la ciudad de Donostia-San Sebastián en un día soleado y a la hora de comer"

Cabe recalcar que el análisis descrito sobre la situación anterior ha debido de ser realizado tanto por el experto en el dominio como por el programador de manera colaborativa y teniendo como soporte la ficha diseñada para tal fin (Sección 4.2).

De esta manera, en sucesivas secciones se explicarán las diferentes funcionalidades de la plataforma, guiadas por el caso de uso expuesto y las diferentes fases que componen la metodología de desarrollo.

5.1. Descripción de la arquitectura de la plataforma

La plataforma *Context Cloud* está compuesta por diferentes módulos que definen su comportamiento y funcionalidades. Cada módulo de la arquitectura se encuentra distribuido en una capa diferente, conformando así las capas de las que se compone la arquitectura de la plataforma, tal y como se propone en el marco teórico descrito en la Sección 3.3.1.

A continuación se muestran las diferentes capas y módulos de los que se compone la arquitectura de la plataforma de manera detallada, así como las diferentes relaciones e interacciones entre los mismos. También, en la figura se reflejan los diferentes actores (programadores o expertos en el dominio) que interactúan con cada una de las capas definidas.

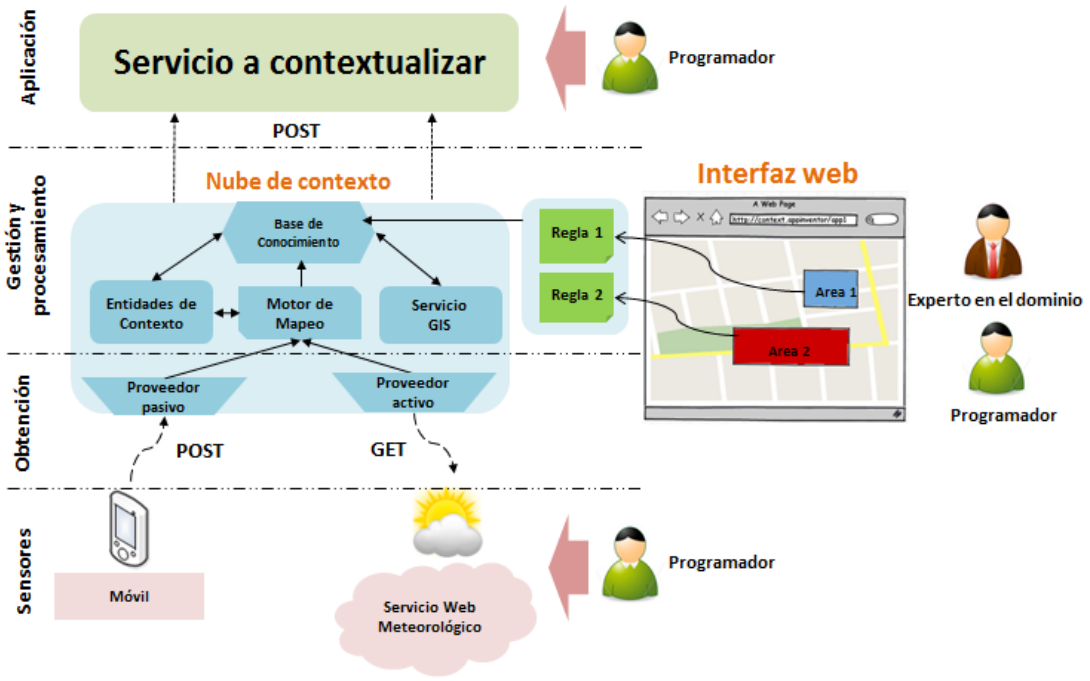


Fig. 28. Arquitectura de la plataforma *Context Cloud*

Los componentes que se identifican en la arquitectura de la plataforma se describen de la siguiente manera.

- *Sensores o fuentes de contexto*: son las entidades que proporcionan los datos de contexto necesarios para identificar situaciones de alto nivel. Estas fuentes pueden estar distribuidas y proporcionar diferentes tipos de datos. Como ejemplo, figuran un dispositivo móvil, de donde se podrían obtener datos como la localización del usuario, y un servicio web meteorológico, de donde se podría obtener el valor de la temperatura en un determinado lugar. Como ya se ha comentado en la Sección 4.1.2 relativa a la metodología de desarrollo, estos sensores pueden requerir de la intervención del programador para adaptar la forma de acceso a los mismos y que la plataforma pueda ser capaz de obtener los datos requeridos.

- *Nube de contexto*: es el módulo encargado de gestionar los datos de contexto, desde su obtención hasta su procesamiento, transformación e interpretación. Aquí se aglutinan por lo tanto, las capas de *obtención*, *gestión y procesamiento* de la información de contexto. La nube de contexto puede ser configurada tanto por el programador como por el experto en el dominio, mediante la interfaz web ofrecida por *Context Cloud*. Esta configuración implica la creación de proveedores de contexto, la creación de un modelo de datos basado en entidades de contexto (Persona, Ciudad, etc.), la creación de áreas y tipos de áreas, y la configuración de reglas asociadas a dichas áreas para modelar las situaciones requeridas de las entidades. Todas estas funcionalidades vienen soportadas por los siguientes módulos.
    - *Proveedores de contexto*: son los encargados de obtener los datos de contexto de las fuentes identificadas. Estos proveedores pueden ser configurados mediante la interfaz web de *Context Cloud*. La obtención de la información de contexto se basa en las buenas prácticas de la Web de las Cosas, ya comentadas en la Sección 3.3.1 y que conforman uno de los requerimientos identificados en el marco teórico del presente trabajo de investigación. Los proveedores se clasifican en dos tipos: pasivos y activos.
        - *Pasivo*: un proveedor de este tipo es el que está a la espera de que las propias fuentes de contexto identificadas envíen de manera proactiva los datos de contexto necesarios. De esta manera, las fuentes de contexto son las que realizan el envío de los datos al proveedor correspondiente. Un ejemplo puede ser un servicio móvil programado para enviar la localización GPS del usuario a la plataforma cada cierto tiempo de manera autónoma mediante peticiones de tipo POST, soportadas por el protocolo HTTP.
        - *Activo*: un proveedor activo es el que accede cada cierto tiempo a una fuente de contexto identificada con el fin de obtener los datos de contexto. Un ejemplo puede ser un servicio web meteorológico expuesto por terceros que es accedido cada hora por un proveedor activo configurado en

la plataforma con el fin de conseguir los datos de contexto referentes a la temperatura de una localización determinada, realizando peticiones de tipo GET soportadas por el protocolo HTTP.

- *Servicio GIS*: la plataforma cuenta con un sistema GIS con el que se pueden gestionar las áreas y tipos de áreas que se crean en la plataforma con el fin de identificar las regiones en el espacio donde se pueden detectar las situaciones requeridas. Este servicio está conectado a un servicio de mapas[36] ofrecido por la plataforma sobre el que se pueden dibujar polígonos con el fin de representar las áreas. A su vez, estas se almacenan sobre una base de datos geoespacial *PostgreSQL*[37] que permite realizar operaciones relativas a la localización de las entidades respecto a las áreas configuradas. Así, se puede saber, por ejemplo, si una determinada entidad cuya localización es conocida se encuentra dentro de una de las áreas definidas en el sistema.

- *Entidades de contexto*: mediante la interfaz web se pueden crear las diferentes entidades de contexto y propiedades que conforman el modelo de datos requerido. Ejemplos de entidades pueden ser Persona, con sus respectivas propiedades como nombre, edad, latitud, longitud, etc. o Ciudad con propiedades como nombre, temperatura, humedad, etc. Todas estas entidades creadas se almacenan en el módulo de entidades de contexto.

- *Motor de mapeo*: es el encargado de realizar las transformaciones de los datos provenientes de las fuentes de contexto y adaptar los mismos al modelo de datos definido en la plataforma. Esta operación se denomina mapeo de los datos. Para tal fin, la plataforma dispone de diálogos de configuración web con los que se puede especificar la correspondencia o mapeo entre los datos obtenidos o recibidos desde las fuentes de contexto y las

[36] https://developers.google.com/maps/?hl=es Último acceso 29-03-2012.
[37] http://www.postgresql.org.es/ Último acceso 29-03-2012.

propiedades definidas en las entidades de contexto que forman parte del modelo de datos.

- *Base de conocimiento*: es el módulo que almacena en memoria todos los datos de contexto obtenidos y convertidos a las entidades de contexto correspondientes. Además, almacena las reglas de comportamiento definidas mediante la interfaz web que sirven para detectar situaciones concretas asociadas a las áreas geográficas configuradas. Cuando se lanzan las reglas definidas debido a que se cumplen las condiciones especificadas sobre la información de contexto recibida, se desencadenan salidas de información de alto nivel que los programadores pueden utilizar posteriormente para contextualizar los sistemas a implementar. Estas salidas pueden dirigirse a puntos de acceso web. Estos datos de salida se representan en formato XML y se encapsulan en peticiones de tipo POST donde posteriormente pueden ser procesadas por los sistemas para adaptar su comportamiento.

- *Servicio a contextualizar*: aquí se encuentra la capa de *aplicación*, donde reside el servicio o sistema que se quiere contextualizar en función de las salidas con información de alto nivel generadas por la plataforma. Tal y como se ha descrito, la plataforma puede enviar información de contexto de alto nivel al servicio o sistema externo cada vez que se detecte alguna de las situaciones definidas y configuradas.

La arquitectura de la plataforma se ha diseñado con el fin de que se puedan especificar distintas configuraciones relativas a sistemas sensibles al contexto diferentes. Para ello, cada sistema a desarrollar puede disponer de su propia nube de contexto en la plataforma donde se gestionarán de manera independiente los datos de contexto necesarios para desarrollar cada sistema.

Así, el programador, que será el actor que disponga de la herramienta, puede crear en la plataforma tantas nubes de contexto como sistemas se vayan a implementar. Esta división de la información de contexto favorece la independencia de datos entre diferentes desarrollos con lo que datos de diferentes sistemas o servicios no interfieren entre ellos.

5.2. Descripción de la interfaz y funcionalidades generales

En esta sección se describen tanto la interfaz de usuario de la plataforma como las funcionalidades de carácter general que se ofrecen para dar soporte a los desarrollos que se realicen con la misma.

5.2.1. Interfaz principal

Context Cloud es una plataforma web, por lo que puede ser accedida mediante cualquier navegador, presentándose la interfaz de usuario que se muestra en la siguiente figura.

Fig. 29. Interfaz web de *Context Cloud*

En esta interfaz de usuario se distinguen diferentes zonas con información relativa a las configuraciones que se permiten realizar mediante la plataforma. Estas configuraciones servirán para determinar y poder detectar las situaciones relevantes para el sistema sensible al contexto a desarrollar.

Se diferencian diferentes zonas en la interfaz con las que se pueden realizar diferentes configuraciones.

- *Tipos de áreas*: gestión de los tipos de áreas.
- *Áreas*: gestión de las áreas de cada tipo creado.
- *Mapa*: visualización y creación de las áreas donde se pueden detectar las situaciones requeridas.

- *Reglas*: configuración de las reglas de comportamiento para modelar las situaciones a detectar.
- *Proveedores activos*: configuración de los proveedores activos.
- *Proveedores pasivos*: configuración de los proveedores pasivos.
- *Entidades*: creación del modelo de datos basado en entidades de contexto y propiedades asociadas a las mismas.

En base a las diferentes configuraciones permitidas por la plataforma, tanto programadores como expertos en el dominio, pueden gestionar la información de contexto necesaria para el funcionamiento del sistema final a implementar.

5.2.2. Funcionalidades generales

En esta sección se detallan las funcionalidades generales que ofrece la plataforma para poder iniciar la gestión de la información de contexto. Así, el acceso a la plataforma se realiza mediante usuario y contraseña, previo registro en la misma. Por lo tanto, los programadores pueden disponer de nubes de contexto y configuraciones independientes en función de los sistemas o servicios que se vayan a implementar. El registro se realiza proporcionando los siguientes datos, tal y como se muestra en la Figura 30.

- Nombre de usuario *(Username)*.
- Contraseña *(Password)*.
- Dirección de correo electrónico *(Email)*.

Fig. 30. Registro de usuario

En caso de disponer de usuario y contraseña, se podrá realizar el ingreso en la misma mediante la pantalla de *login* o de acceso de usuario que se muestra a continuación.

Fig. 31. Acceso de usuario

Una vez realizado el registro de usuario, se muestra un diálogo con información para el **programador**, que será necesaria para el proceso de obtención de información de contexto y configuración de las fuentes de contexto.

- *Identificador de nube (Cloud id)*: este identificador se utiliza para saber a qué nube tienen que enviar los datos las fuentes de contexto identificadas.
- *Clave de desarrollador (Developer key)*: esta clave sirve para autentificar la fuente de contexto que envía de manera proactiva datos a la nube.

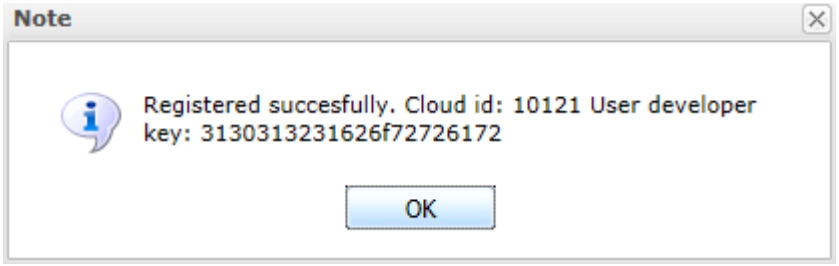
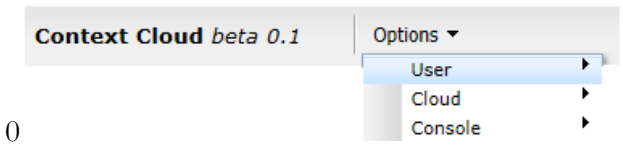


Fig. 32. Datos del registro de usuario

En la parte superior de la interfaz de la plataforma se encuentra el menú de opciones. Aquí se pueden encontrar las opciones relativas a la gestión del usuario registrado en la plataforma *(User)*, de su nube de contexto *(Cloud)* y la consola de depuración que ofrece la plataforma *(Console)*, tal y como se muestra en la figura que sigue a continuación.



0

Fig. 33. Menú de opciones

Estas opciones principales se dividen a su vez en otra serie de opciones que se describen a continuación.

- *Usuario (User)*
  - *Salida (Logout)*: salida del sistema.
  - *Información (Information):* muestra la información del usuario de la plataforma, donde se incluyen los siguientes datos:
    - Nombre de usuario *(Username)*.
    - URL *(Uniform Resource Locator)* de la nube de contexto donde las fuentes registradas tienen que enviar los datos. Está compuesta por el dominio de la plataforma y el identificador de la nube de contexto del usuario.
    - Clave de desarrollador *(Key)*.

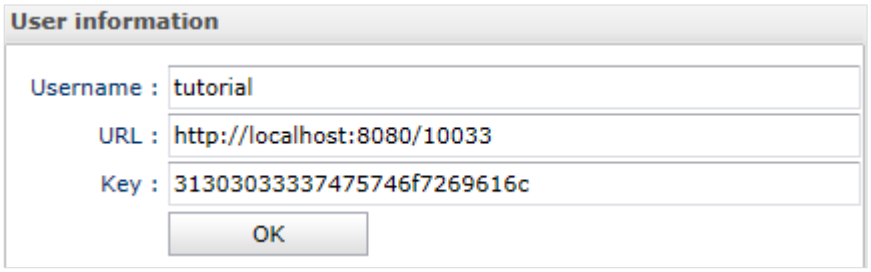


Fig. 34. Información de usuario

- *Nube (Cloud)*
  - *Preparada (Ready)*: comprueba si la nube de contexto está activa y preparada para recibir información de contexto.
  - *Vaciar (Clear)*: vacía la base de conocimiento borrando todos los datos de contexto almacenados en la nube.
- *Consola (Console)*
  - *Mostrar (Show)*: muestra la consola de depuración.
  - *Ocultar (Hide)*: oculta la consola de depuración.

La consola de depuración muestra diferentes mensajes internos de la plataforma. Esta consola está pensada para dar soporte en la fase de configuración de la plataforma definida en la metodología (Capítulo 4). De esta manera se pueden depurar las configuraciones realizadas y se pueden encontrar fallos en las mismas con mayor facilidad.

En la siguiente figura se muestra un ejemplo de diferentes mensajes que pueden aparecer en la consola de depuración, referentes a procesos internos en la gestión de la información de contexto.

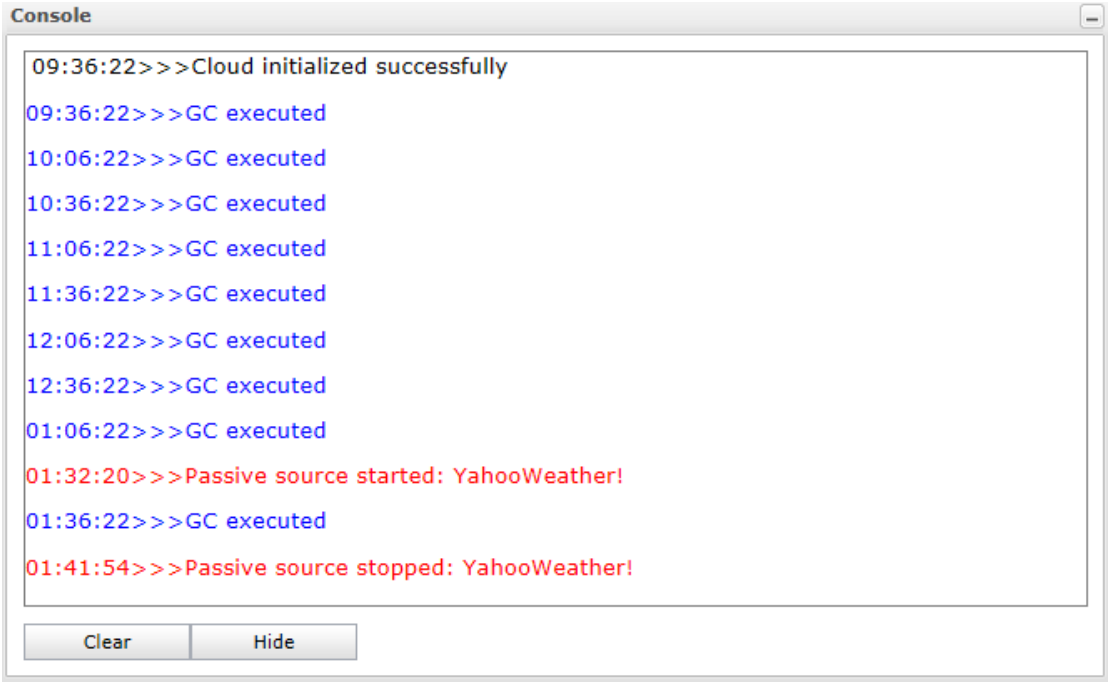


Fig. 35. Consola de depuración

Estos mensajes de información se muestran en diferentes colores, en función del módulo que genere dichos mensajes.

- *Azul*: generados por la base de conocimiento.
- *Rojo*: generados por los proveedores de contexto.
- *Verde*: generados por las reglas.
- *Negro*: generados por la nube de contexto en general.

La consola dispone de dos botones.

- *Limpiar (Clear):* limpia los mensajes en consola.
- *Ocultar (Hide):* oculta la consola.

5.3. Fuentes de contexto

Una vez realizada la fase de análisis donde se identifican y especifican las diferentes situaciones, la información de contexto necesaria y las fuentes de donde poder obtener dicha información, viene la fase de configuración de las propias fuentes de contexto. En esta fase, es el programador el que tiene que identificar y analizar las fuentes de contexto, ya que muy posiblemente, se requiera algún tipo de implementación para

adecuar las fuentes de datos y posibilitar así el envío de información de contexto a la plataforma y/o que la misma plataforma pueda acceder a las fuentes de datos a por la información de contexto.

Es importante tener en cuenta que para que un proveedor configurado en la plataforma pueda acceder a las fuentes de contexto o pueda recibir información de las mismas, éstas deben cumplir una serie de requisitos indispensables, basados en las buenas prácticas establecidas por la comunidad de la Web de las Cosas (Guinard et al., 2011). En estas buenas prácticas se establecen las formas de interacción, basadas en el protocolo HTTP, que deben implementarse con el fin de obtener información de contexto de las fuentes de datos identificadas. A continuación se muestran las características que deben cumplir las fuentes de datos para poder ser accedidas por la plataforma o para que las propias fuentes puedan enviar datos a la plataforma.

- La fuente de contexto tiene que estar conectada a Internet.
- Para que la plataforma pueda obtener los datos de contexto, debe ofrecer la posibilidad de recibir peticiones de tipo GET.
- Si la propia fuente de contexto va a enviar la información a la plataforma, deberá ser capaz de realizar peticiones de tipo POST a la plataforma, enviando la información de contexto en la propia petición.

Además, se establecen una serie de restricciones referentes al formato de los datos obtenidos de las fuentes de contexto.

- La información ofrecida por las fuentes de contexto debe estar en formato XML[38].
- El esquema XML de los datos obtenidos debe ser siempre el mismo. Es decir, tanto el número de etiquetas y atributos del XML como sus nombres, deben ser siempre los mismos. La única variación que puede existir entre diferentes envíos de información son los valores de las etiquetas y atributos del documento XML que almacenan los datos de contexto.

[38] La plataforma puede ser extendida para poder procesar otros modelos de datos comúnmente utilizados basados en formato JSON.

Las restricciones impuestas por la plataforma se muestran en la Figura 36, donde se ha representado un servicio web meteorológico que hará las veces de fuente de contexto para ilustrar el ejemplo. Sobre esta fuente de contexto, se ha configurado un proveedor activo que obtiene los datos de contexto cada cierto tiempo (t) mediante peticiones de tipo GET en las que se codifica que la información meteorológica que se quiere obtener es relativa a la ciudad de Donostia. Así, en el primer acceso (t=0), el servicio devuelve un documento XML que contiene el valor de la temperatura (etiqueta *temperature*) y las unidades de la misma (etiqueta *units*). En el instante t=1, se vuelve a obtener una medición, donde el valor de la temperatura ha variado. Sin embargo, en el instante t=2, el documento XML contiene una nueva etiqueta llamada *humidity*. Esta modificación en el esquema del documento obtenido propiciará que la plataforma no procese el documento XML.

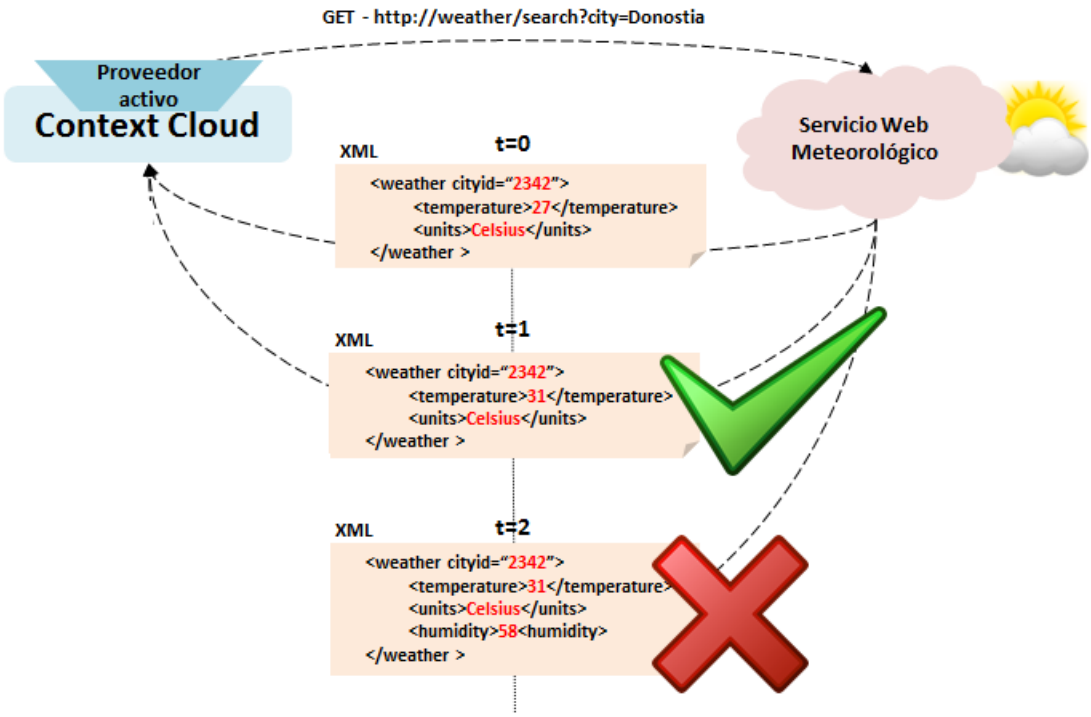


Fig. 36. Restricciones de las fuentes accedidas por los proveedores activos

Siguiendo con el ejemplo propuesto para ilustrar el funcionamiento de la plataforma y en relación a la ficha de análisis realizada en la descripción de la metodología (Sección 4.2), se van a considerar como fuentes de contexto, el dispositivo móvil del usuario (GPS y guía móvil) y un servicio web meteorológico.

Las fuentes de contexto que se encuentran en el dispositivo móvil necesitan una capa software adicional con el fin de poder enviar los datos de contexto a la plataforma.

Para ello, el programador tiene que implementar un servicio asociado a la guía móvil que el visitante se descarga, que sea capaz de enviar desde el propio dispositivo móvil a la plataforma, la información de contexto requerida, en este caso, la localización (GPS), el identificador del visitante (número de teléfono) y el origen del mismo (guía móvil).

El servicio se pondrá en funcionamiento cada vez que el visitante utilice la guía. Este servicio puede programarse para enviar la localización cada dos minutos mediante peticiones POST, ya que al ser una información de contexto dinámica y variable, es interesante que la plataforma tenga las máximas referencias posibles del dato. Además, el servicio tiene que enviar mediante peticiones de tipo POST el identificador y el origen del usuario cada vez que este entre en la guía móvil o realice algún cambio sobre la configuración de su perfil de usuario.

Además, el programador tiene que identificar algún servicio web meteorológico para obtener la temperatura de la ciudad de Donostia-San Sebastián, como por ejemplo el ofrecido por *Yahoo Weather*[39]. Este servicio proporciona las condiciones meteorológicas de una ciudad determinada pasada como parámetro mediante peticiones de tipo GET.

Puede suceder que la fuente de información de contexto no ofrezca la posibilidad de ser accedida directamente por la plataforma o que haya datos que no se adapten a las restricciones requeridas y necesiten algún tipo de transformación previa antes de que los datos puedan ser obtenidos por la plataforma. En ese caso, el programador puede implementar un acceso intermedio entre la fuente de datos y la plataforma, con el fin de ofrecer los mecanismos de acceso y formato de datos necesarios para proporcionar la información de contexto pertinente.

5.4. Proveedores de contexto

El siguiente paso de la metodología es la configuración por parte del programador de los proveedores de contexto con el fin de que obtengan o reciban los datos de las fuentes identificadas. Como ya se ha comentado anteriormente, los proveedores proporcionados por la plataforma desarrollada pueden ser tanto activos como pasivos, en función de si son éstos los que acceden a la fuente de contexto directamente o es la propia fuente de contexto la que envía los datos al proveedor.

[39] http://developer.yahoo.com/weather/ Último acceso 29-03-2012.

Siguiendo con el caso de uso propuesto, se va a mostrar la configuración de un proveedor activo para acceder al servicio web meteorológico identificado y un proveedor pasivo, que esperará a que se envíe la información de contexto del visitante desde el servicio instalado en la guía móvil del dispositivo. A continuación se describe la configuración de cada uno de estos proveedores.

5.4.1. Proveedor activo

El proveedor activo tiene que ser configurado para que pueda acceder a la fuente de contexto de manera periódica. Esta configuración tiene que llevarse a cabo por el programador, ya que como se ha comentado con anterioridad, puede requerir de ajustes en las propias fuentes de contexto accedidas para que el proveedor funcione correctamente.

Así, se va a configurar un acceso a la fuente de datos meteorológicos ofrecida por *Yahoo Weather*, donde se expone una API a la que se pueden lanzar peticiones GET con el objetivo de obtener el tiempo meteorológico de cualquier ciudad del mundo. En este caso se obtendrá el tiempo meteorológico de la ciudad de Donostia-San Sebastián.

Para ello, se ha configurado una URL sobre la que se realizarán las peticiones GET por parte del proveedor.

http://weather.yahooapis.com/forecastrss?p=SPXX0026&u=c[40]

Con esta petición, se obtiene un documento XML donde figuran todos los datos de los que dispone el servicio respecto al tiempo meteorológico de la ciudad. Para configurar un proveedor activo en la plataforma, se necesitan proporcionar los siguientes datos.

- *Nombre (Name)*: nombre del proveedor.
- *Descripción (Description)*: descripción del proveedor.
- *URL*: dirección de la fuente de contexto sobre la que se realizará la petición de tipo GET para obtener los datos.
- *Segundos* (*Seconds)*: período en segundos que indicará el intervalo de tiempo entre los diferentes accesos a la fuente de contexto.

[40] Último acceso 15-08-2012.

- *ID particular (Custom ID)*: mediante este campo se puede asignar un valor que sirva como identificador de los datos obtenidos de la fuente de contexto. Este campo puede ser vacío si en el documento XML devuelto por la fuente ya existiera algún valor que pudiera actuar como identificador de la información recibida.

En la siguiente figura se muestran los datos utilizados en la configuración del proveedor.

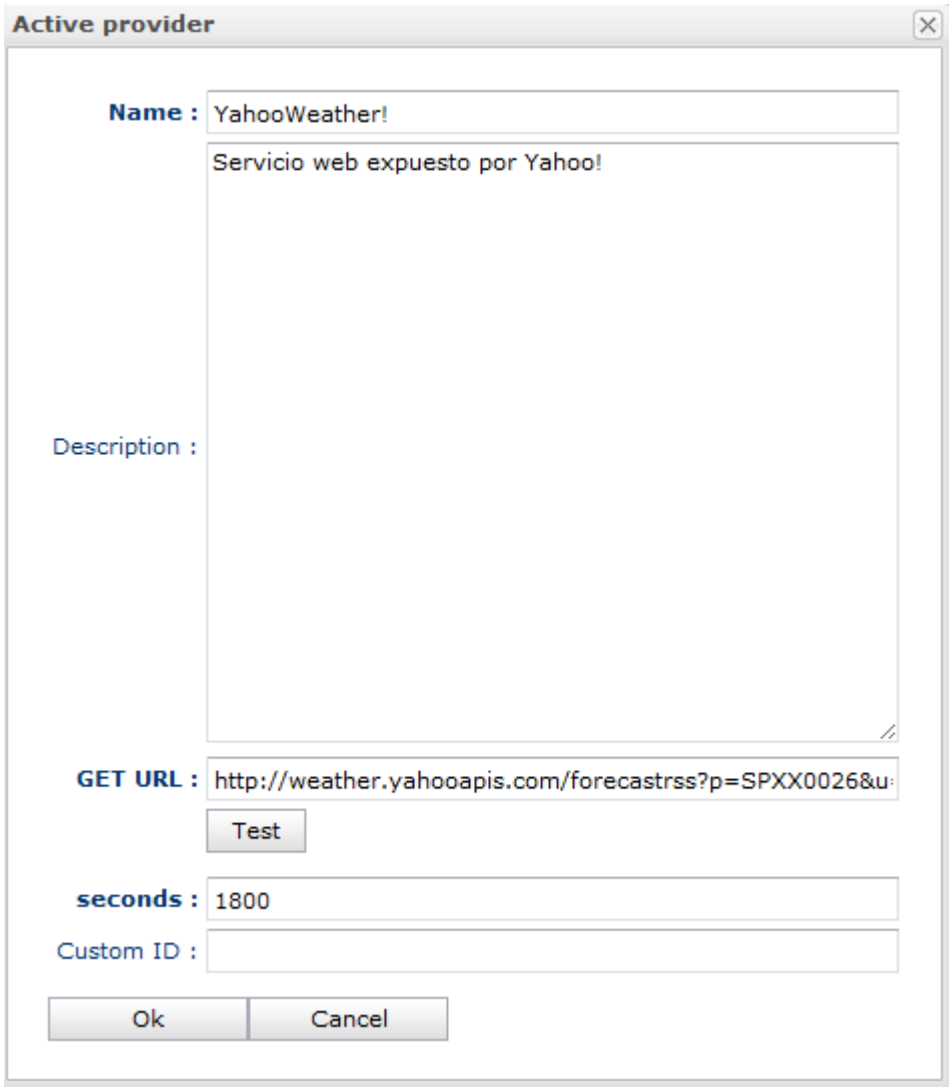


Fig. 37. Configuración de un proveedor activo

En este caso se ha establecido un intervalo de media hora (1800 segundos) entre cada acceso a la fuente de datos. El campo *Custom ID* se ha dejado vacío, ya que se va a utilizar la etiqueta que contiene el nombre de la ciudad como identificador de los datos provenientes del servicio web meteorológico.

Antes de confirmar la configuración del proveedor en la plataforma es necesaria su validación mediante el botón de testeo "Test". Al pulsar el botón, el proveedor lanza la petición al servicio web, extrayendo y procesando los datos provenientes del documento XML de respuesta.

De esta manera, es el propio proveedor el que se encarga de extraer los datos del documento XML y formatearlos para que puedan ser interpretados de una manera más sencilla por los usuarios de la plataforma. Así, tras pulsar el botón "Test", se muestran al usuario de la plataforma los datos extraídos en un cuadro de diálogo mediante pares del tipo "nombre, valor". Estos pares son extraídos de todos los elementos y atributos del documento XML con sus respectivos valores. Además, los pares se formatean mediante los caracteres ">>>" en función del nivel jerárquico de dichos elementos en el documento XML.

Un extracto del aspecto del cuadro de diálogo que muestra el detalle de los datos obtenidos de la fuente de contexto es el que se presenta en la siguiente figura, donde se muestran todos los elementos, atributos y valores del documento que contiene información meteorológica para la ciudad de Donostia-San Sebastián.

Property	Value
rss	
rss-version	2.0
rss-xmlns:geo	http://www.w3.org/2003/01/geo/wgs84_pos#
rss-xmlns:yweather	http://xml.weather.yahoo.com/ns/rss/1.0
>>>channel	
>>>>>>title	Yahoo! Weather - Donostia-San Sebastian, SP
>>>>>>link	http://us.rd.yahoo.com/dailynews/rss/weather/Donostia-San_Sebastian__SP/*htt
>>>>>>description	Yahoo! Weather for Donostia-San Sebastian, SP
>>>>>>language	en-us
>>>>>>lastBuildDate	Fri, 30 Mar 2012 11:00 am CEST
>>>>>>ttl	60
>>>>>>yweather:location	
>>>>>>yweather:location-city	Donostia-San Sebastian
>>>>>>yweather:location-country	SP
>>>>>>yweather:location-region	
>>>>>>yweather:units	
>>>>>>yweather:units-distance	km
>>>>>>yweather:units-pressure	mb
>>>>>>yweather:units-speed	km/h
>>>>>>yweather:units-temperature	C
>>>>>>yweather:wind	
>>>>>>yweather:wind-chill	13
>>>>>>yweather:wind-direction	300
>>>>>>yweather:wind-speed	4.83
>>>>>>yweather:atmosphere	

Fig. 38. Datos obtenidos por el proveedor activo

Para ilustrar la forma en la que se interpretan y muestran los datos de la figura anterior, se va a describir el tratamiento realizado sobre un extracto del documento XML devuelto por el servicio meteorológico, el cual se muestra a continuación.

```
<rss xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0"
        xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#" version="2.0">
    <channel>
        <title>Yahoo! Weather - Donostia-San Sebastian, SP</title>
        <link>
            http://us.rd.yahoo.com/dailynews/rss/weather/Donostia-San_Sebastian__SP/
            *http://weather.yahoo.com/forecast/SPXX0026_c.html
        </link>
```

Fig. 39. Extracto de documento XML

Teniendo en cuenta los datos del extracto del documento XML, se puede ver cómo la raíz del documento es un elemento de nombre *rss*, que a su vez dispone de diferentes atributos, como son *yweather*, *geo* y *version*. Este elemento raíz se muestra en el diálogo de la interfaz pero sin ningún valor asociado. Además, aparecen todos sus atributos junto con sus valores. Para identificar que estos atributos pertenecen al elemento *rss*, se les ha añadido a modo de prefijo dicho elemento, formando así los campos *rss-version*, *rss-xmlns:geo* y *rss-xmlns:yweather* respectivamente.

A continuación se encuentra el elemento *channel*, que es hijo del primer elemento *rss*, por lo que se incluyen los caracteres "**>>>**" para mostrar esta jerarquía. A su vez, este elemento tiene como hijos los elementos *title* y *link*. Esta jerarquía, por lo tanto, vendrá expresada mediante el conjunto de caracteres "**>>>>>>**". La disposición anteriormente explicada se refleja en la siguiente figura.

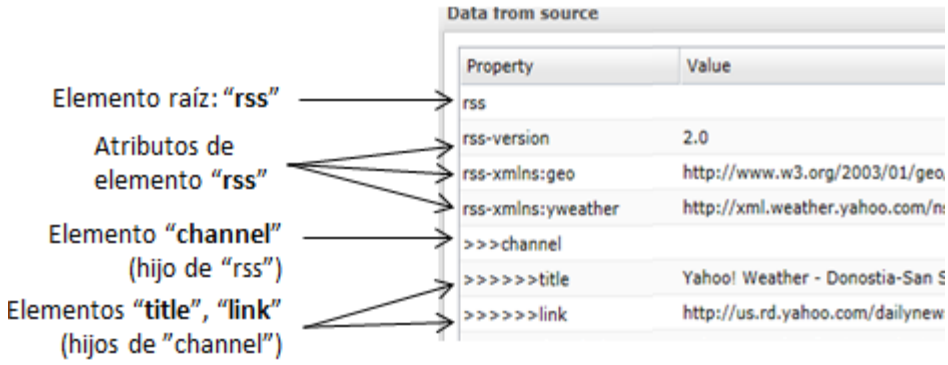


Fig. 40. Visualización de los datos extraídos de la fuente de contexto

Una vez configurado y probado el proveedor de la plataforma, aparecerá en el listado de proveedores activos, tal y como se muestra en la siguiente figura.

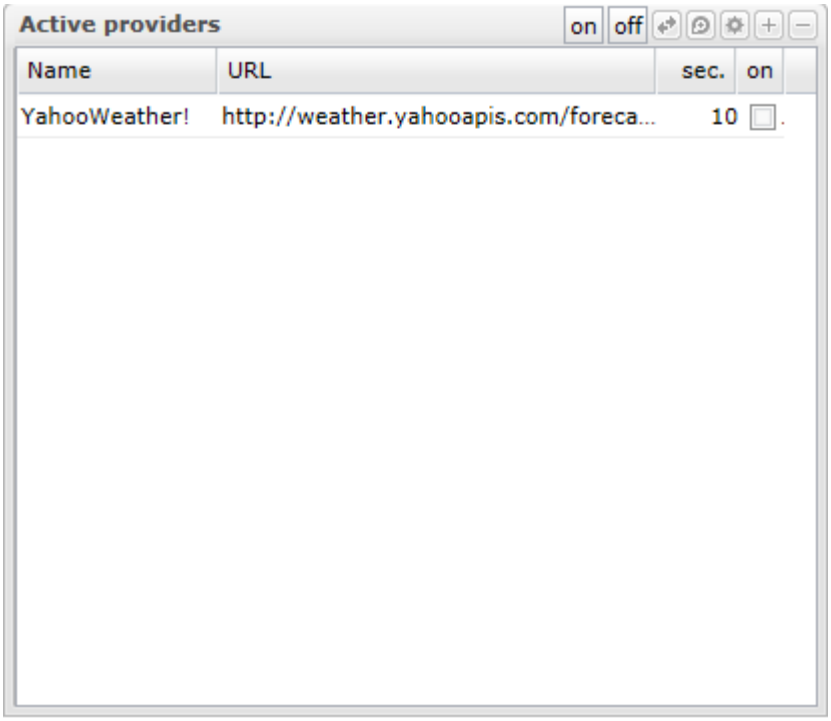


Fig. 41. Listado de proveedores activos

El siguiente paso a la configuración del proveedor, es la activación del mismo para que empiece a obtener los datos de la fuente identificada y estos sean procesados por la plataforma. Para ello se utilizan los controles on off con los que se puede activar o desactivar la obtención de datos del proveedor sobre la fuente de contexto. Existen además otros controles que permiten gestionar los proveedores activos.

- *Activar/Desactivar* on off: posibilita la activación o desactivación del proveedor activo de la fase de obtención de información de contexto de las fuentes identificadas.
- *Mapear*: posibilita la configuración de los mapeos entre los datos obtenidos y el modelo de datos. Esta funcionalidad se describe posteriormente en la Sección 5.7.
- *Visualizar*: muestra el detalle de la información del proveedor activo y los últimos datos obtenidos de la fuente de contexto.
- *Modificar*: posibilita la modificación de los datos de un proveedor activo.
- *Crear*: posibilita la creación de un nuevo proveedor activo.

- *Borrar* : borra el proveedor activo seleccionado.

Un ejemplo de la ventana de detalle del proveedor activo que se obtiene al pulsar el botón de visualización es el que se muestra en la siguiente figura. En este cuadro de diálogo se muestran los parámetros de configuración del proveedor junto con los últimos datos obtenidos de la fuente de contexto.

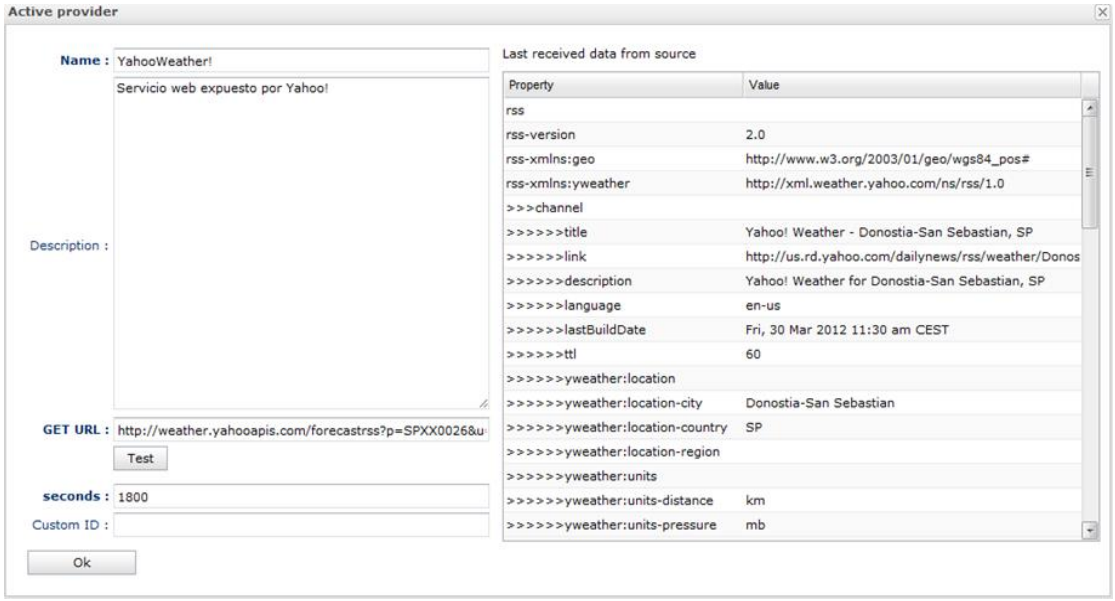


Fig. 42. Detalle de un proveedor activo

5.4.2. Proveedor pasivo

Los datos provenientes de la guía instalada en el dispositivo móvil serán enviados por un servicio software implementado en la propia guía, que se encargará de obtener el identificador de usuario junto con los datos de localización cada dos minutos para enviarlos a la plataforma. De esta manera se conocerá la localización del visitante de una manera detallada. Además, el servicio obtendrá el origen del visitante extrayéndolo de su perfil configurado en la guía y lo enviará a la plataforma en cada acceso a la aplicación junto con el identificador de usuario, que en este caso será el número de teléfono del visitante.

Para que la plataforma reciba los datos enviados por la fuente de contexto, en este caso la guía móvil, deben ser configurados dos proveedores pasivos en la plataforma. Un primer proveedor, de nombre "Movil-GPS" que recibirá el identificador del usuario, que en este caso corresponderá con el número de teléfono utilizado en el registro de la guía móvil, y los datos de localización (latitud y longitud) obtenidos del dispositivo GPS.

El segundo proveedor, de nombre "Movil-perfil", recibirá el identificador de usuario junto con el origen del visitante obtenido del perfil configurado en la propia guía.

En la Figura 43 se muestra el proceso de creación de uno de los proveedores mencionados anteriormente, concretamente el "Movil-GPS", donde se ha especificado la siguiente información.

- *Nombre (Name)*: nombre del proveedor pasivo.
- *Descripción (Description)*: descripción del proveedor pasivo.
- *ID Particular* (*Custom ID)*: mediante este campo se puede asignar un valor que sirva como identificador de los datos obtenidos de la fuente de contexto. Este campo puede ser vacío si en el documento XML ya existiera algún valor que pudiera actuar como identificador.

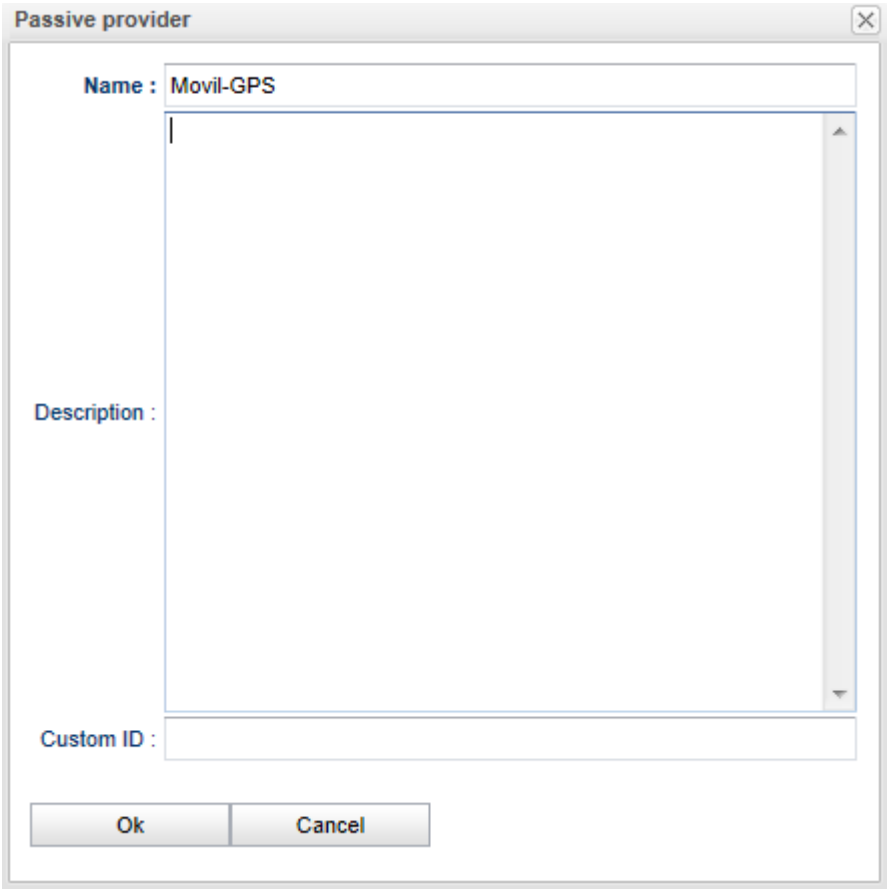


Fig. 43. Creación de un proveedor pasivo

Una vez creados los dos proveedores, saldrán en el listado de proveedores pasivos, donde se ofrecen diversos controles para gestionar los proveedores creados.

- *Mapear* : posibilita la configuración de mapeos entre los datos obtenidos y el modelo de datos. Esta funcionalidad se describe en la Sección 5.7.
- *Visualizar* **:** muestra la información del proveedor pasivo y los últimos datos obtenidos de la fuente de contexto.
- *Modificar* **:** posibilita la modificación de los datos de un proveedor pasivo.
- *Crear* **:** posibilita la creación de un nuevo proveedor pasivo.
- *Borrar* **:** borra el proveedor pasivo seleccionado.

La siguiente figura muestra el cuadro de control de los proveedores pasivos creados.

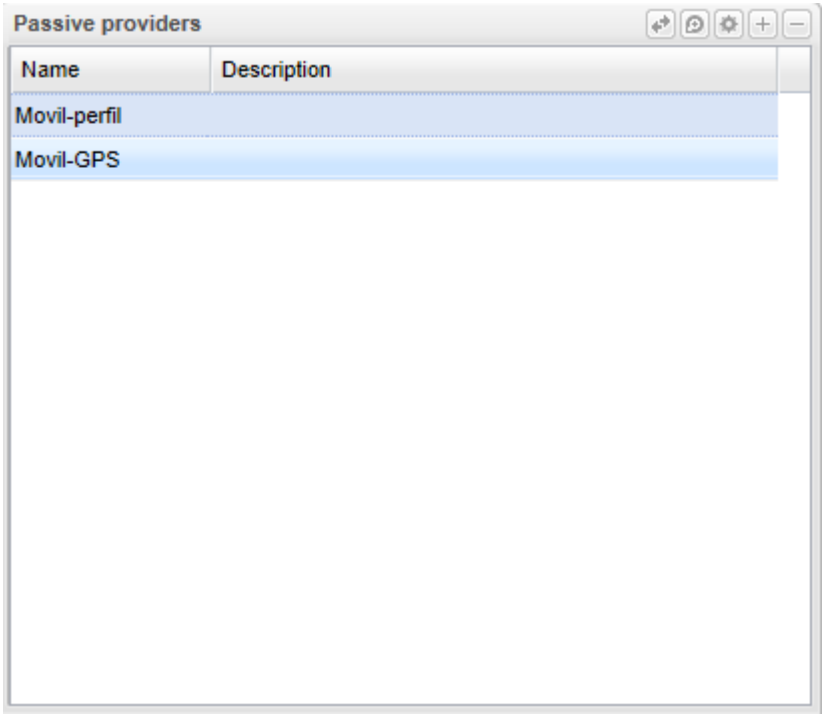


Fig. 44. Listado de proveedores pasivos

Para que las fuentes de contexto puedan enviar la información a los diferentes proveedores configurados, es necesaria una URL accesible en la plataforma a la que las fuentes de contexto puedan hacer peticiones de tipo POST que contengan la información de contexto en formato XML. La dirección donde enviar dicha información se conforma de la siguiente manera.

URL_plataforma/**id_nube**/providers/passive/**nombreproveedor**/data?developer-key=**X**

Como se puede observar, hay **tres parámetros** que deben ser especificados en la URL resultante, que son el identificador de la nube de contexto, el nombre del proveedor configurado en la plataforma y la clave de desarrollador. Un ejemplo de URL resultante donde poder enviar la información de contexto en formato XML mediante una petición POST al proveedor pasivo anteriormente creado "Movil-GPS" sería el que se muestra a continuación.

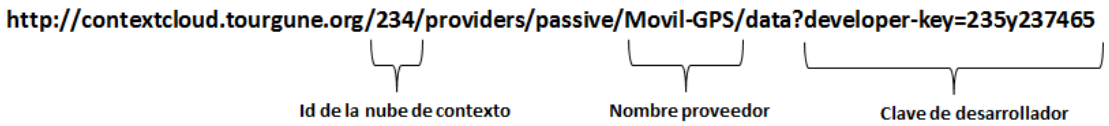


Fig. 45. URL para el envío de información de contexto a la plataforma

Cabe destacar que la información de contexto que es enviada por la guía instalada en el dispositivo móvil a los dos proveedores configurados tiene un dato en común, que es el *identificador* del visitante o usuario de la guía móvil. Mediante este identificador del visitante (número de teléfono), la plataforma es capaz de agregar los datos relativos al mismo usuario pero provenientes de diferentes fuentes de información (dispositivo GPS del móvil proporcionando localización y perfil configurado en la guía móvil descargada por el visitante).

De esta manera, la plataforma es capaz de aglutinar o unificar automáticamente diferentes datos de contexto, provenientes de diferentes fuentes de contexto o sensores que hagan referencia a la misma entidad de contexto. En este caso, la plataforma aglutinará la información referente al mismo usuario, en función del identificador recibido por la misma.

Una vez configurados los dos proveedores de contexto pasivos, las fuentes pueden empezar a enviar datos en formato XML sobre la URL generada de manera proactiva, utilizando para ello peticiones HTTP de tipo POST. En la figura a continuación se muestra un ejemplo de visualización de los últimos datos recibidos por el proveedor "Movil-GPS", donde se muestran los valores referentes al identificador, la latitud y la longitud, además del detalle del propio proveedor.

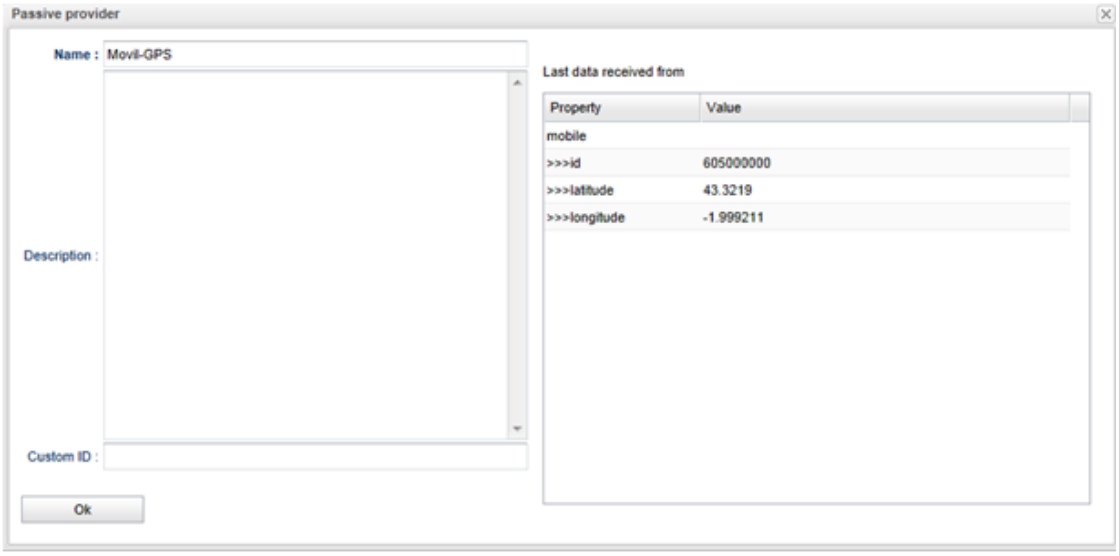


Fig. 46. Detalle de un proveedor pasivo

5.5. Áreas y tipos de áreas

El siguiente paso de la metodología es la configuración por parte de programadores y expertos en el dominio de las áreas y tipos de áreas donde pueden detectarse las situaciones. Esta funcionalidad de la plataforma está estrechamente relacionada tanto con la propia definición de situación propuesta en el presente trabajo de investigación, como con lo comentado en las Secciones 3.3.2 y 3.3.6, donde se describen tanto la representación espacial necesaria en el modelo de datos manejado por la plataforma, como la gestión de la movilidad de las entidades participantes en las situaciones a detectar.

Así, la plataforma permite la creación de tipos de áreas y áreas donde pueden detectarse las situaciones a configurar. La plataforma ofrece un servicio de mapas sobre el que se pueden visualizar y crear las áreas en base a polígonos dibujados sobre el propio mapa. Para la implementación de esta funcionalidad de mapas se ha utilizado el servicio *Google Maps*[41].

Para poder crear una nueva área tiene que estar asociada a un tipo de área determinada. Por lo tanto, el primer paso es la creación de un nuevo tipo de área. Para el caso de uso concreto que se está tratando en esta descripción de funcionalidades, se va a crear un tipo de área de nombre "Ciudad" al que pertenecerá el área Donostia-San

[41] https://developers.google.com/maps/?hl=es Último acceso 15-08-2012.

Sebastián que se creará posteriormente. Este tipo de área permitiría configurar diferentes áreas relativas a diferentes ciudades si fuera necesario.

Para crear un tipo de área nueva, hay que pulsar el control e introducir los datos requeridos, tal y como se muestra en la Figura 47.

- *Nombre (Name)*: nombre del tipo de área.
- *Color (Color)*: el color elegido identificará a todas las áreas creadas del tipo creado sobre el mapa.
- *Descripción (Description)*: descripción del tipo de área.

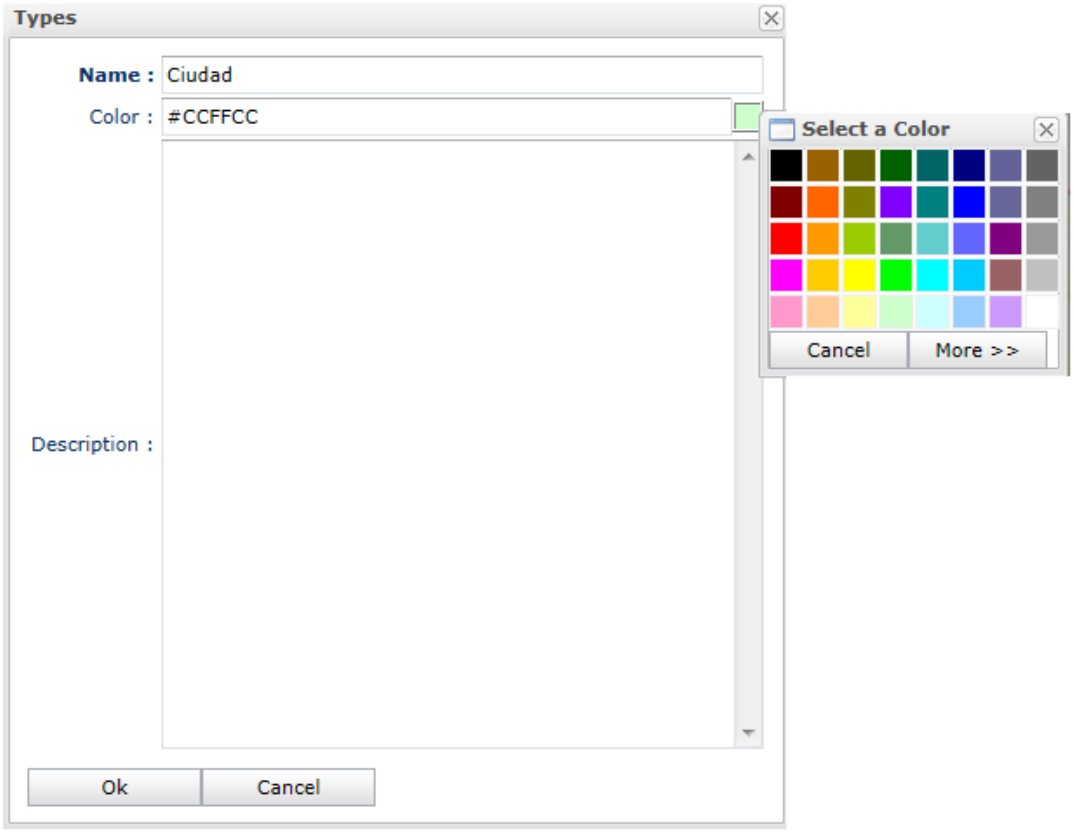


Fig. 47. Creación de un tipo de área nuevo

Una vez creado el tipo de área, aparecerá en el listado de tipos de áreas, junto con la descripción y el color elegido para representar dicho tipo de área, tal y como se muestra en la siguiente figura.

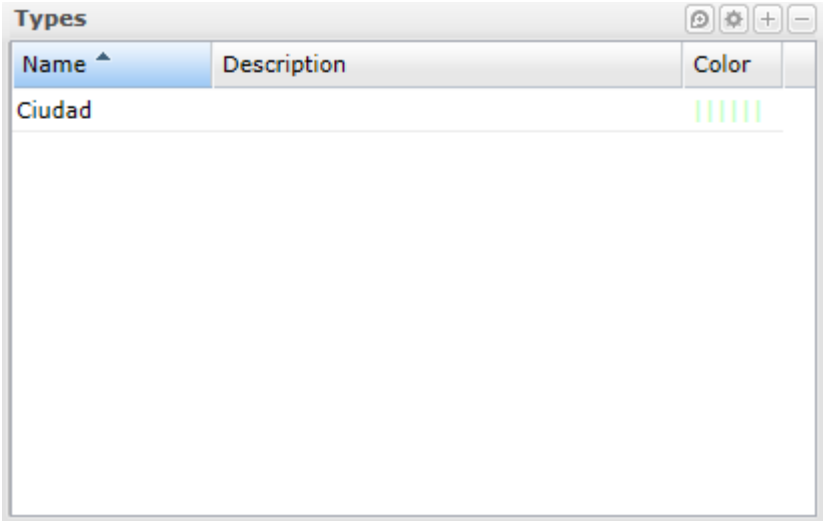


Fig. 48. Listado de los tipos de áreas

Además, se ofrecen diferentes controles para gestionar los tipos de áreas. A continuación se detallan cada uno de estos controles.

- *Visualizar* : muestra los datos del tipo de área seleccionado.
- *Modificar* : posibilita la modificación de los datos de un tipo de área.
- *Crear* : posibilita la creación de un nuevo tipo de área.
- *Borrar* : borra el tipo de área seleccionado. Al borrar un tipo de área se borrarán también las áreas creadas de dicho tipo y las reglas asociadas a las áreas afectadas.

Una vez creado un tipo de área es cuando se pueden crear las áreas asociadas al mismo. En este caso se va a crear un área para identificar la ciudad de Donostia-San Sebastián, que es donde debe estar el visitante para que pueda ser detectada la situación del ejemplo. Para ello, se debe seleccionar el tipo de área "Ciudad" y se debe pulsar sobre el control para crear una nueva área, donde se completarán los datos requeridos, tal y como se muestra en la Figura 49.

- *Nombre (Name)*: nombre del área.
- *Descripción (Description)*: descripción del área.

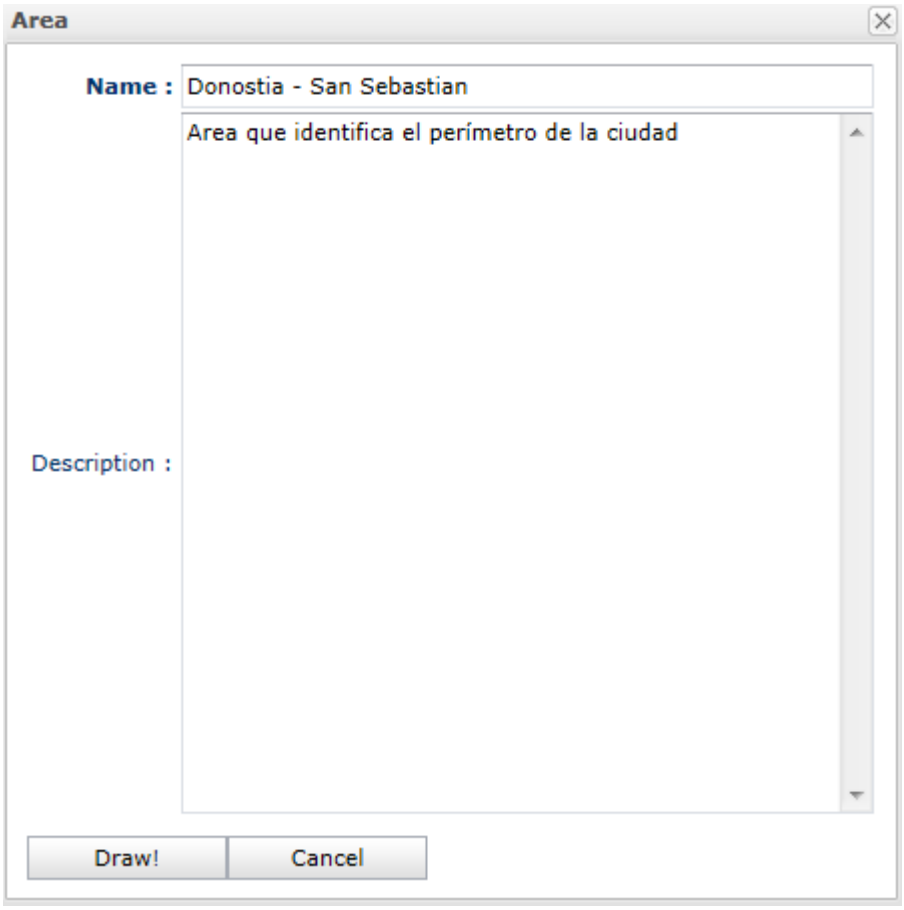


Fig. 49. Creación de una nueva área

Una vez introducidos los datos del área, se posibilitará la creación del polígono que delimite el área sobre el mapa proporcionado mediante el control "Dibujar" *("Draw")*. En la siguiente figura se muestra sobre el mapa el área dibujada que delimita la ciudad de Donostia-San Sebastián.

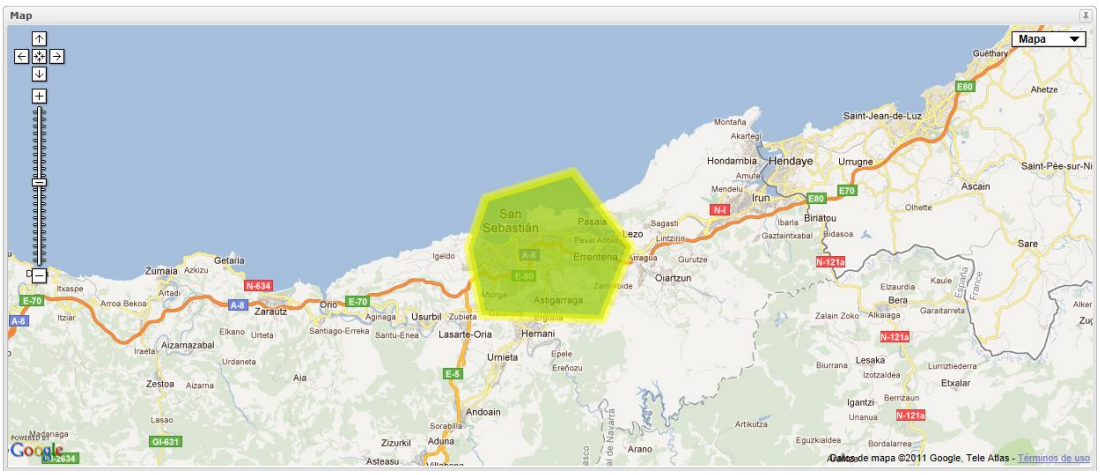


Fig. 50. Polígono que define el área creada

Este cuadro de mapas contiene además un control  con el que se puede fijar la posición del mapa y el zoom, para que sea recordado en posteriores sesiones.

Una vez que se ha creado el área, se muestra en el listado de áreas junto con su descripción.

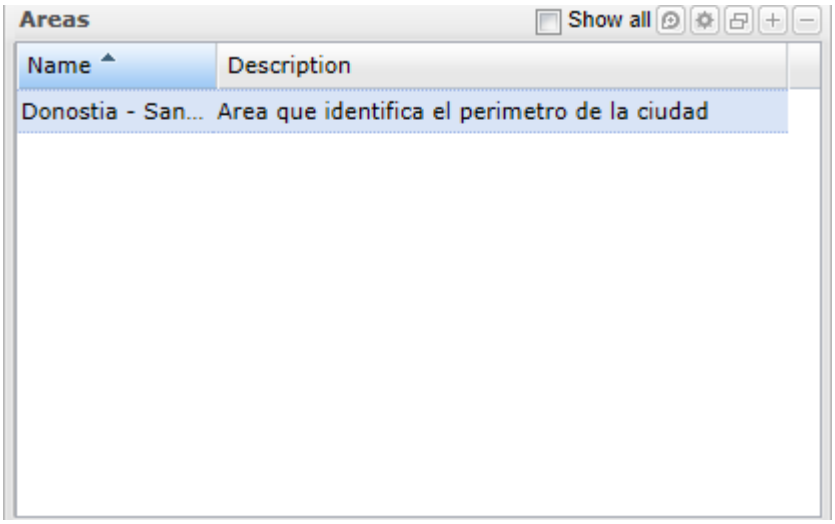


Fig. 51. Listado de áreas

Además, se proporcionan diferentes controles para gestionar las áreas configuradas en la plataforma.

- *Mostrar todas*  : muestra todas las áreas creadas pertenecientes a todos los tipos de área existentes.
- *Visualizar*  : muestra los datos del área seleccionada.
- *Modificar datos*  : posibilita la modificación de los datos del área seleccionada.
- *Modificar polígono*  : posibilita la modificación del polígono que define el área seleccionada.
- *Crear*  : posibilita la creación de una nueva área del tipo de área seleccionado.
- *Borrar*  : borra el área seleccionada y todas sus reglas asociadas.

De esta manera se configura la dimensión espacial que se tiene en consideración para detectar la situación en curso.

5.6. Modelo de datos

El siguiente paso de la metodología es la creación de un modelo de datos que sirva para representar y almacenar toda la información de contexto que llega a la plataforma desde las diferentes fuentes identificadas y a través de los proveedores creados en la misma. Este modelo de datos toma como referencia las entidades definidas en la fase de análisis y puede ser creado de manera colaborativa entre programador y experto en el dominio, aunque es el programador el que tendrá un mayor peso debido a su familiarización con este tipo de tareas de modelado.

Continuando con el ejemplo concreto que se está utilizando en este documento para mostrar las funcionalidades de la plataforma, se tienen que crear dos entidades de contexto para conformar el modelo de datos que dará apoyo a la configuración de la situación a tratar. Estas entidades son "Persona" y "Ciudad".

Por una parte se tiene que crear la entidad "Persona" con el fin de representar la información referente al visitante y usuario de la guía móvil. Esta entidad es una entidad primaria, es decir, que la situación es detectada sobre el comportamiento de la misma. Por otra parte, se tiene que crear la entidad "Ciudad" con el fin de representar la información referente a las condiciones meteorológicas de la ciudad.

Para crear una entidad de contexto, existe el control  , el cual despliega un diálogo de configuración que se muestra en la Figura 53, donde se tienen que completar los siguientes datos.

- *Nombre (Name)*: nombre de la entidad.
- *Descripción (Description)*: descripción de la entidad.
- *Propiedades (Properties)*: propiedades que componen la entidad.

A continuación se muestra la creación de la entidad "Persona". Para ello, hay que crear la entidad y añadir posteriormente las propiedades, mediante el control "Añadir" *("Add")*. En este caso, se tienen que crear las propiedades necesarias para almacenar la información de contexto de interés que proviene de las fuentes de datos. Así, se tienen que crear las propiedades para representar el número de teléfono del visitante, que hará las veces de identificador, la latitud, la longitud y el origen.

Cabe destacar que cada entidad creada tiene que tener obligatoriamente una propiedad que actúe como identificador de la misma. Esta propiedad sirve para que la

plataforma pueda agregar en una misma instancia de entidad datos provenientes de diferentes fuentes de contexto referentes a la misma instancia de entidad.

En este caso, se va a crear la propiedad "num_telefono" para que actúe como identificador de la entidad "Persona". En la propia pantalla de configuración de la propiedad, se puede especificar si actúa o no como identificador, además del tipo de dato que almacenará. En esta ocasión la propiedad se va a marcar de tipo entero *(Integer)*. Los posibles tipos de datos de una propiedad son los siguientes.

- Entero *(Integer)*
- Texto *(String)*
- Verdadero/falso *(Boolean)*
- Decimal *(Float)*
- Lista *(List)*
- Decimal de doble precisión *(Double)*
- Fecha *(Date)*

La siguiente figura muestra el cuadro de diálogo donde se crea la propiedad que actúa de identificador de la entidad "Persona".

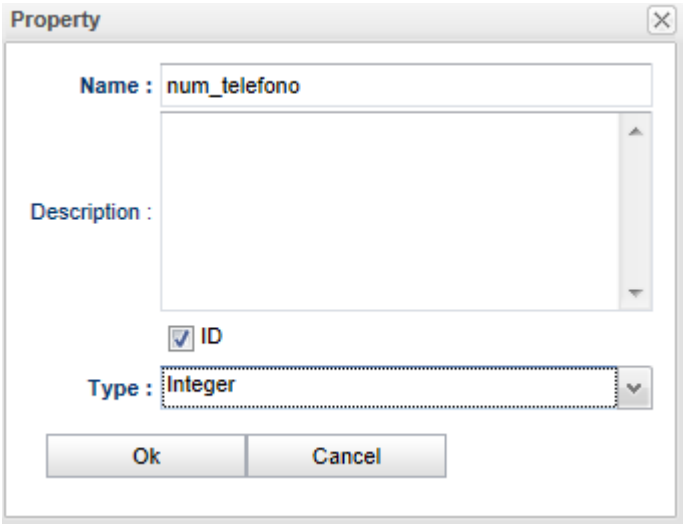


Fig. 52. Creación de una nueva propiedad

Además, las entidades que se crean pueden ser marcadas como "geo-referenciadas" *("Georeferenced Entity")*. En el caso de que una entidad se considere como geo-referenciada, el sistema añadirá automáticamente las propiedades "latitud" *("latitude")* y

"longitud" *("longitude")* a la entidad, y serán gestionadas por el sistema GIS de la plataforma para saber si una entidad de este tipo está dentro de un área que se haya creado anteriormente.

En este caso, la entidad "Persona" tiene que ser marcada de tipo geo-referenciada, ya que se debe controlar la localización del visitante con el fin de conocer si se encuentra en la zona geográfica referente a la ciudad de Donostia-San Sebastián. Estas propiedades relativas a la localización del visitante almacenarán los datos proporcionados por el GPS del dispositivo móvil. Este almacenamiento se configura mediante el proceso de transformaciones que se describe en la Sección 5.7.

El cuadro de configuración resultante para la entidad "Persona" y todas sus propiedades se muestra en la Figura 53.

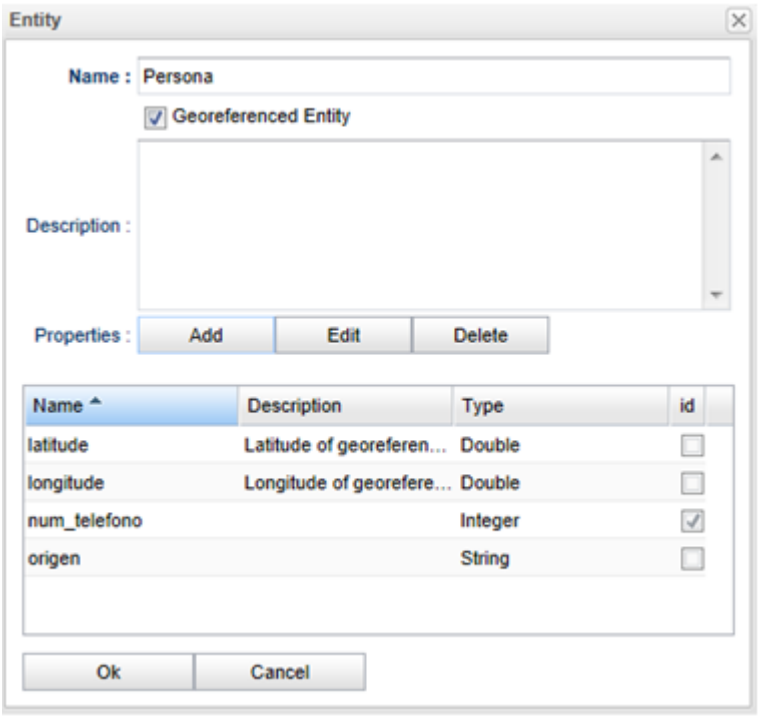


Fig. 53. Creación de la entidad "Persona"

Por su parte, se tiene que crear la entidad "Ciudad" con el fin de almacenar los datos referentes al tiempo meteorológico de la ciudad de Donostia-San Sebastián. Las propiedades de las que consta la entidad son el "nombre" de la ciudad, el cual actuará como identificador de la propia entidad, y la "temperatura" de la misma. Estos datos son relevantes para detectar la situación a tratar, tal y como se refleja en el análisis de la misma. La configuración completa de la entidad se muestra en la siguiente figura.

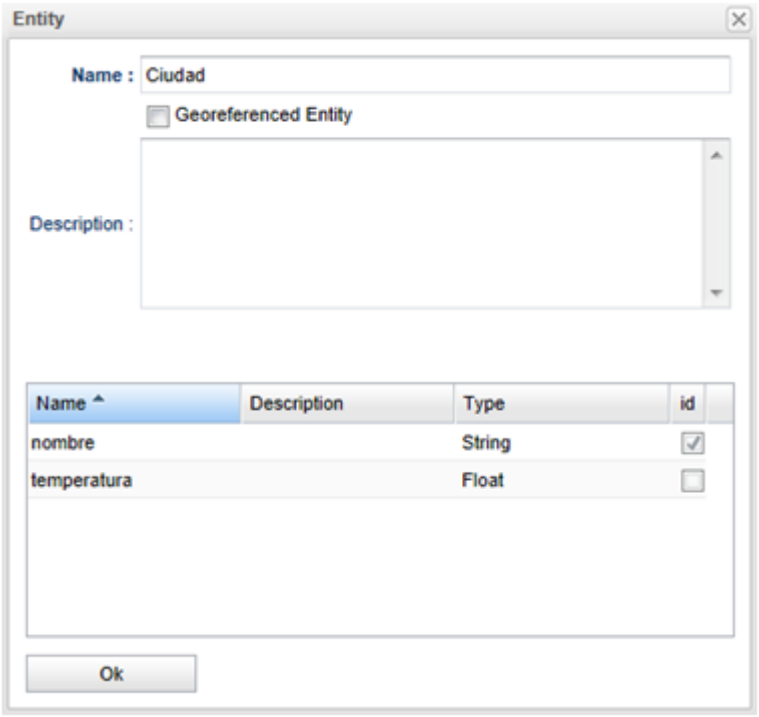


Fig. 54. Configuración de la entidad "Ciudad"

Una vez creadas las entidades necesarias, se muestran en el listado, tal y como se detalla en la siguiente figura.

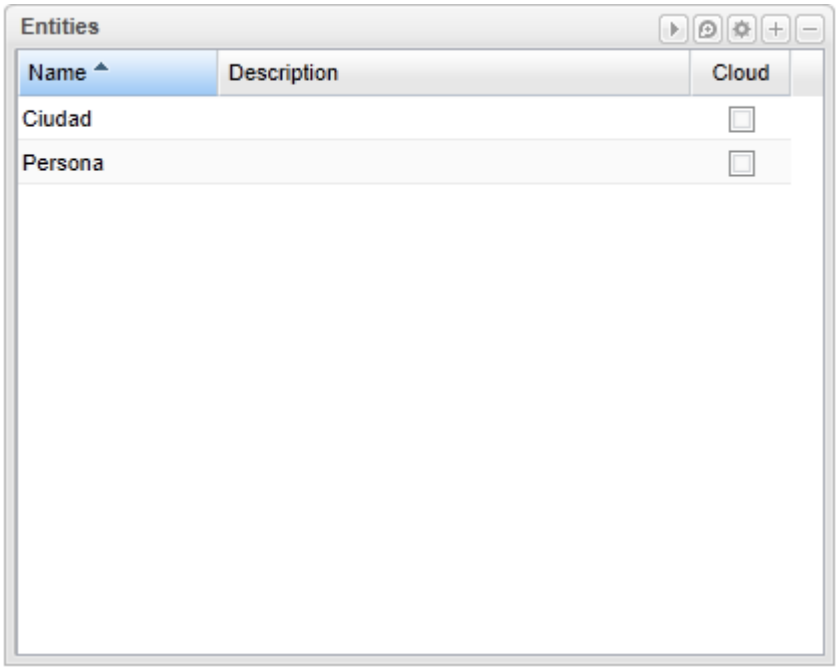


Fig. 55. Gestión de entidades de contexto pertenecientes al modelo de datos

El siguiente paso a la creación de las entidades que conforman el modelo de datos es el despliegue de las mismas en la nube de contexto con el fin de que la plataforma

pueda gestionarlas. Para realizar dicho despliegue se tiene que utilizar el control . Este control transforma la especificación de las entidades en clases de tipo *Java Bean*[42] con el fin de ser gestionadas por la plataforma de manera interna. Esta transformación y gestión de las clases se realiza de forma automática y transparente al usuario de la plataforma.

La columna "Nube" *("Cloud")* que aparece en el listado de entidades indica si una entidad está presente o no en la nube de contexto del usuario de la plataforma. Solamente las entidades que estén en la nube podrán ser gestionadas por la misma. Esta acción de despliegue en la nube se debe realizar cada vez que se produzca alguna modificación en el listado de entidades.

Además existen diferentes controles para gestionar las entidades de contexto. Las acciones posibles para gestionar dichas entidades son las que se detallan a continuación.

- *Desplegar entidades en la nube* **:** cada vez que se crea una entidad nueva o una entidad existente es modificada, el campo "*Cloud"* del listado aparece desmarcado. Esto significa que esa entidad no está presente en la nube de contexto. Para que la entidad sea desplegada y la plataforma pueda gestionarla, es necesario hacer un despliegue. Una vez desplegadas las entidades, se muestran como desplegadas en la nube mediante la columna *"Cloud"*, tal y como se muestra en la siguiente figura.

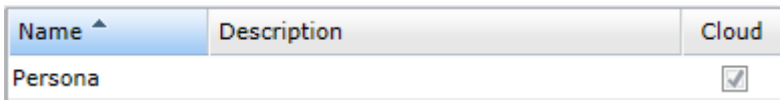


Fig. 56. Entidad de contexto desplegada en la nube gestionada por la plataforma

- *Visualizar* : visualiza los datos de la entidad seleccionada y sus propiedades.
- *Modificar* : permite la modificación de los datos de la entidad y de sus propiedades.
- *Crear* **:** permite la creación de entidades y sus propiedades.
- *Borrar* **:** borra la entidad seleccionada.

[42] http://docs.oracle.com/javase/tutorial/javabeans/index.html Último acceso 22-08-2012.

5.7. Transformaciones

El siguiente paso es la configuración de las transformaciones necesarias para que los datos provenientes de las fuentes de contexto puedan ser almacenados en el modelo de datos definido. En este caso, las transformaciones que ofrece la plataforma son mapeos o correspondencias entre los datos de contexto recibidos y las propiedades de las entidades definidas en el modelo de datos. Este proceso de configuración de mapeos puede realizarse también por el programador y/o el experto en el dominio.

Por cada proveedor creado pueden configurarse los mapeos correspondientes para los datos de contexto recibidos u obtenidos de las fuentes de contexto. Para acceder a la herramienta de configuración de mapeos, la plataforma proporciona el control en las secciones relativas a los listados de los proveedores tanto activos como pasivos.

En el diálogo de mapeo se muestran ciertos controles para configurar las correspondencias de datos. Así, en la parte izquierda se sitúan los controles para gestionar los mapeos y en la parte derecha se muestran los últimos datos obtenidos o recibidos por el proveedor. Los controles para gestionar los mapeos permiten añadir una nueva configuración de mapeo *("Add")*, modificar *("Edit")* y/o borrar una ya existente *("Delete")*.

En la siguiente figura se muestra la pantalla de configuración de mapeos para el proveedor activo configurado anteriormente, donde se muestran los últimos datos relativos al tiempo meteorológico de la ciudad de Donostia-San Sebastián obtenidos por el proveedor.

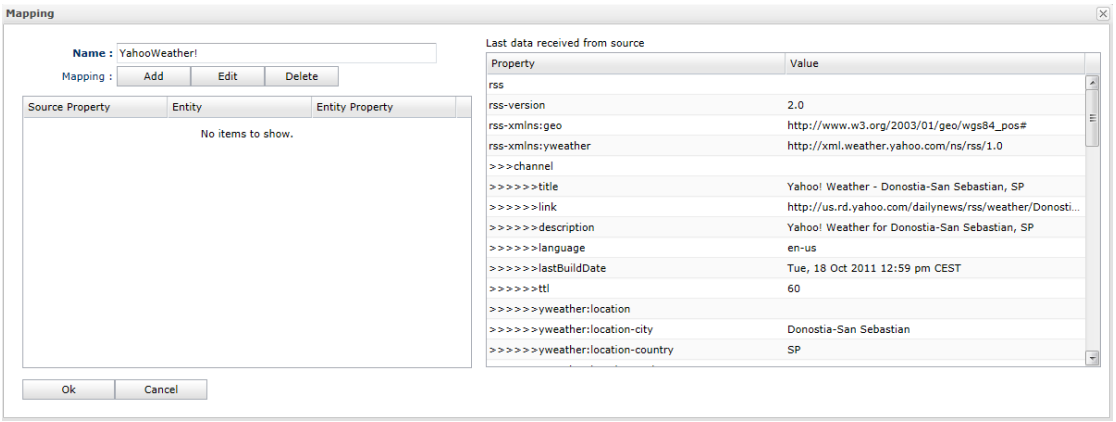


Fig. 57. Pantalla de configuración de mapeos

A continuación, se muestra la configuración de los mapeos entre la entidad de contexto "Ciudad" y el proveedor que obtiene los datos provenientes del servicio web

meteorológico. Para ello, se tiene que pulsar sobre el botón "Añadir" *("Add"),* el cual lanza el cuadro de diálogo que se muestra a continuación.

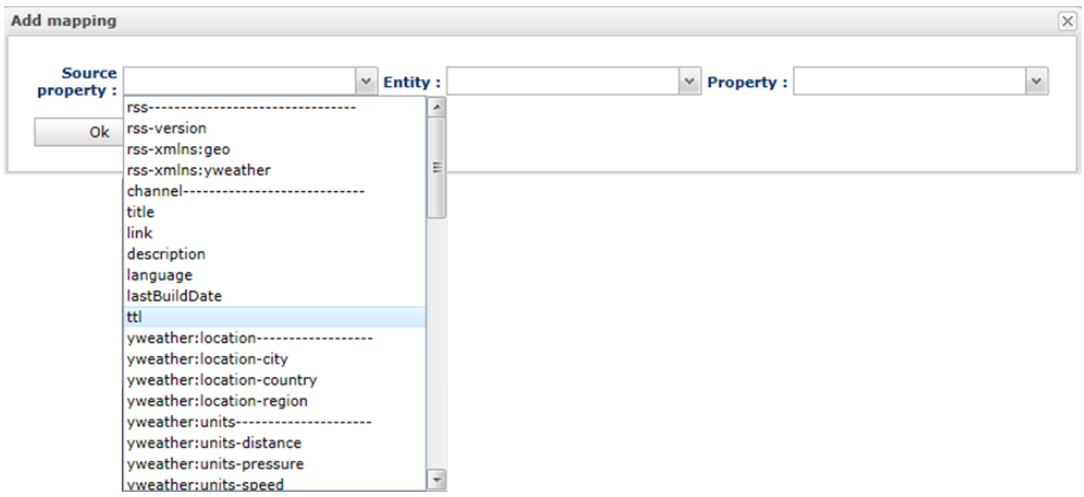


Fig. 58. Configuración de un nuevo mapeo

En el cuadro de diálogo, se posibilita la selección de un dato obtenido de la fuente de contexto, una entidad del modelo de datos que se encuentre desplegada en la nube y una propiedad de dicha entidad con el fin de configurar un nuevo mapeo. De esta manera se permiten identificar los valores de contexto obtenidos de la fuente que se deben almacenar en las propiedades de las diferentes entidades de contexto creadas y que conforman el modelo de datos.

Solamente se pueden elegir aquellas propiedades de la fuente de contexto que no dispongan de los símbolos "----------", ya que estos indican que son propiedades sin valor. Hay que destacar que la plataforma comprueba si el valor de una propiedad de la fuente de contexto puede ser transformado al valor de la propiedad de la entidad que conforma el mapeo. Es decir, si por ejemplo en la propiedad de la fuente de contexto existe un valor de tipo texto *(String)*, éste no podrá ser mapeado a una propiedad de entidad de tipo entero *(Integer)* ya que puede contener valores que no son números.

Además, la plataforma solamente permite la configuración de un mapeo por propiedad de entidad, es decir, que no puede existir una propiedad de una entidad de contexto que esté mapeada con datos de fuentes de contexto diferentes al mismo tiempo. La excepción a esta regla, es la propiedad de la entidad que actúa como identificador, ya que es utilizada para agregar datos de diferentes fuentes de contexto que se refieren a la misma entidad. Por otra parte, los datos recibidos u obtenidos por

un mismo proveedor pueden mapearse sobre propiedades de entidades de contexto diferentes.

Estas restricciones se ilustran en la Figura 59, donde se quiere mapear el dato "Latitud" proveniente de la "Fuente de Contexto 2" a la propiedad de nombre "Lat" perteneciente a la entidad "Persona" que ya estaba mapeada con datos proporcionados por la "Fuente de Contexto 1". La plataforma no posibilita realizar el mapeo de la propiedad con el valor proporcionado por la "Fuente de Contexto 2" ya que existe previamente un mapeo sobre dicha propiedad de la entidad "Persona" que sería machacado por este nuevo valor.

Por su parte, se puede apreciar que la plataforma permite que datos provenientes de diferentes fuentes de contexto pero relativos a la misma propiedad, en este caso la propiedad "Id", sean mapeados, siempre y cuando la propiedad a ser mapeada corresponda con el identificador de la entidad. De esta manera, la plataforma podrá agregar los datos relativos a la latitud, longitud y ritmo cardíaco de la persona con identificador "1" en la misma instancia de entidad.

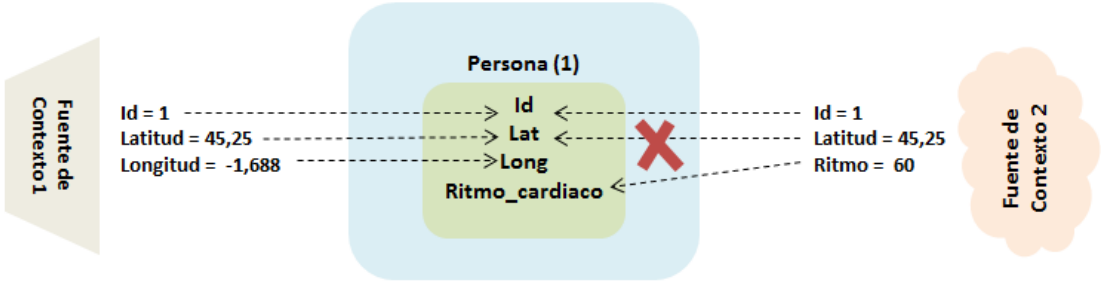


Fig. 59. Restricciones de mapeos

Siguiendo con el caso de uso que guía la descripción de las funcionalidades de la plataforma, se tienen que realizar los mapeos de los datos obtenidos por los proveedores con las entidades y propiedades creadas en el modelo de datos. Así, se deben configurar los mapeos para las entidades "Persona" y "Ciudad".

A continuación se muestra la configuración del mapeo para la entidad "Ciudad" con los datos provenientes del servicio web meteorológico.

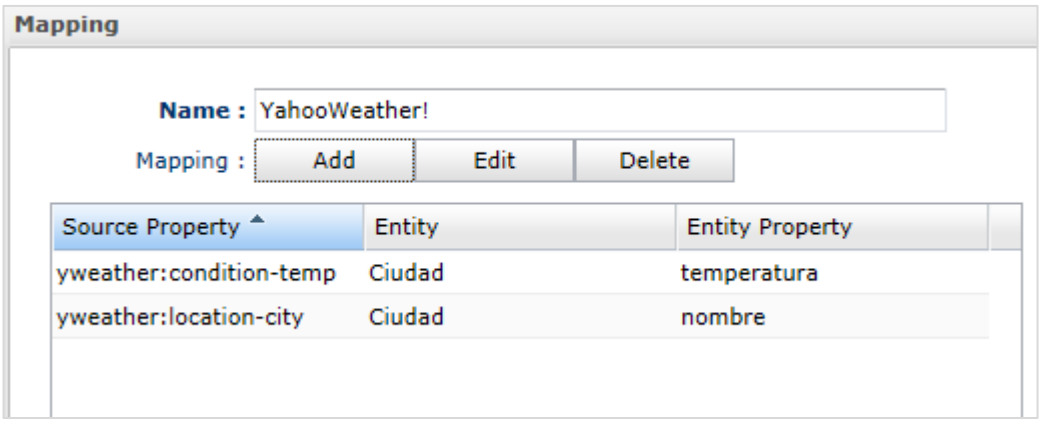


Fig. 60. Mapeos del proveedor meteorológico

Por su parte, los mapeos de la entidad "Persona" tienen que configurarse para los dos proveedores restantes. Por una parte se tienen que mapear los datos referentes a la localización de la persona y por otra parte se deben mapear los datos relativos al perfil de la persona. En este caso, la propiedad "id" de la entidad "Persona" y los mapeos configurados con ambos proveedores, servirán para que la plataforma pueda agregar los datos relativos a la misma instancia de entidad, provenientes de fuentes de contexto diferentes.

En la Figura 61 se muestran los mapeos configurados entre el proveedor "Movil-GPS" y la entidad "Persona".

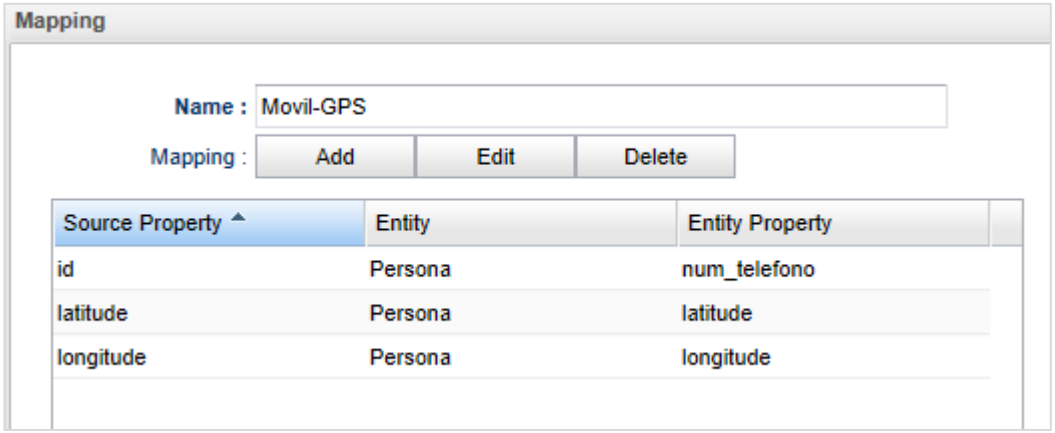


Fig. 61. Mapeos del proveedor "Movil-GPS"

Por su parte, la Figura 62 muestra la configuración de los mapeos entre el proveedor "Movil-perfil" y la entidad "Persona".

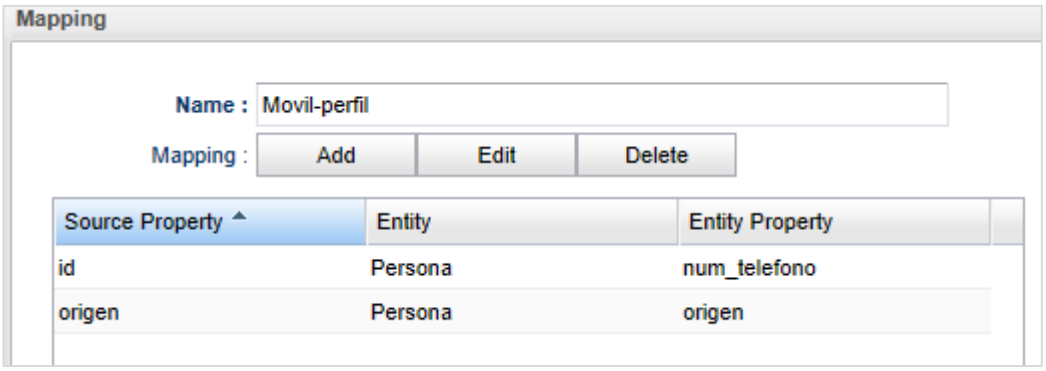


Fig. 62. Mapeos del proveedor "Movil-perfil"

5.8. Reglas

Una vez que se han realizado los pasos anteriores de la metodología, configurando tanto las fuentes de contexto, los proveedores, las áreas, el modelo de datos y las transformaciones necesarias, el siguiente paso es la configuración de las reglas para la detección de las situaciones.

La definición de reglas se basa en el modelo de datos diseñado y en los valores de contexto que el propio modelo almacena, fruto de la obtención de información de contexto por parte de los proveedores. De esta manera, utilizando las condiciones definidas sobre dichos datos anteriormente especificadas en la fase de análisis de cada situación, se pueden configurar las reglas necesarias para detectar la situación en curso.

Como ya se comentó en la Sección 3.3.3, la utilización de reglas lógicas para la definición de las situaciones posibilita que los expertos en el dominio puedan explicitar de antemano su conocimiento mediante la plataforma con el fin de identificar las situaciones relevantes para el sistema a implementar. Además, hay que tener en cuenta que el sistema de razonamiento y las propias reglas deben integrar tanto la posibilidad de gestionar la movilidad de las entidades como el factor temporal de las situaciones, características cruciales en la definición de situación propuesta en el presente trabajo. El sistema de reglas implementado permite así definir condiciones tanto espaciales (determinando si una entidad está en un área definida en la plataforma) como temporales, con el fin de detectar cada situación.

Las reglas son también las encargadas de generar las salidas de alto nivel necesarias para que los sistemas sensibles al contexto que vayan a ser implementados puedan adaptar su comportamiento en función de las mismas. Así, la plataforma sigue el patrón de gestión de contexto definido en la especificación de la plataforma, detallado en la

Sección 3.3.1 relativa a la arquitectura de la misma, donde es la plataforma la que recibe entradas con información de contexto de bajo nivel, procesa la información en base a las reglas definidas y genera salidas de alto nivel con información referente a las situaciones detectadas.

Para la especificación de reglas mediante la plataforma, se ha prestado especial atención a los expertos en el dominio, proporcionando un entorno amigable y automatizado para la creación de reglas, en el que no se necesitan conocimientos demasiado complejos para la creación de las mismas. Además de ofrecer estos mecanismos para la creación de las reglas, se ha optado por proporcionar la posibilidad de editar de forma manual estas reglas. De esta manera, los programadores y los expertos en el dominio con un conocimiento más avanzado en el uso de la plataforma, pueden editar las reglas directamente sin necesidad de utilizar los controles disponibles para tal fin.

El formato de las reglas es de tipo "Evento Condición Acción" *(Event Condition Action - ECA)* (López-de-Ipiña, 2001), es decir, que se especifican las condiciones que se deben cumplir referentes a los datos de contexto gestionados por la plataforma (eventos), especificando también las acciones que deben realizarse cuando se cumplan dichas condiciones. Una de las acciones principales que se ofrecen en la definición de las reglas es la de generar salidas con información referente a la situación detectada. De esta manera se produce un proceso de inferencia de conocimiento que se traduce en información de alto nivel que otros sistemas pueden utilizar para adaptar su comportamiento y funcionalidades.

En siguientes secciones se detallan la arquitectura de la Base de Conocimiento en la que se encuentra el sistema de reglas y los controles para la configuración de las mismas, todo ello en base a la situación de ejemplo propuesta.

5.8.1. Arquitectura de la Base de Conocimiento

En la Base de Conocimiento se almacenan las instancias de las entidades que han sido procesadas por el módulo de mapeo. Concretamente, se almacenan en un espacio en memoria reservado por el motor de reglas, el cual es el principal componente de la Base de Conocimiento.

De esta manera, cuando una nueva información de contexto llega a la plataforma mediante los proveedores configurados, el módulo de mapeo realiza las

transformaciones pertinentes para que esos datos se conviertan en una instancia de una entidad determinada y sea insertada o actualizada en la Base de Conocimiento. El motor de reglas se ha implementado tomando como base el sistema experto proporcionado por el software *Drools Expert*[43].

Sobre el sistema experto utilizado como base se han implementado ciertas capas superiores, con el fin de proporcionar toda la funcionalidad requerida por la plataforma y especificada en los requerimientos identificados con anterioridad. Estas capas proporcionan automatismos clave para la gestión de la información de contexto. La Figura 63 muestra las tres capas implementadas sobre el motor de reglas *Drools Expert*, las cuales conforman la Base de Conocimiento.

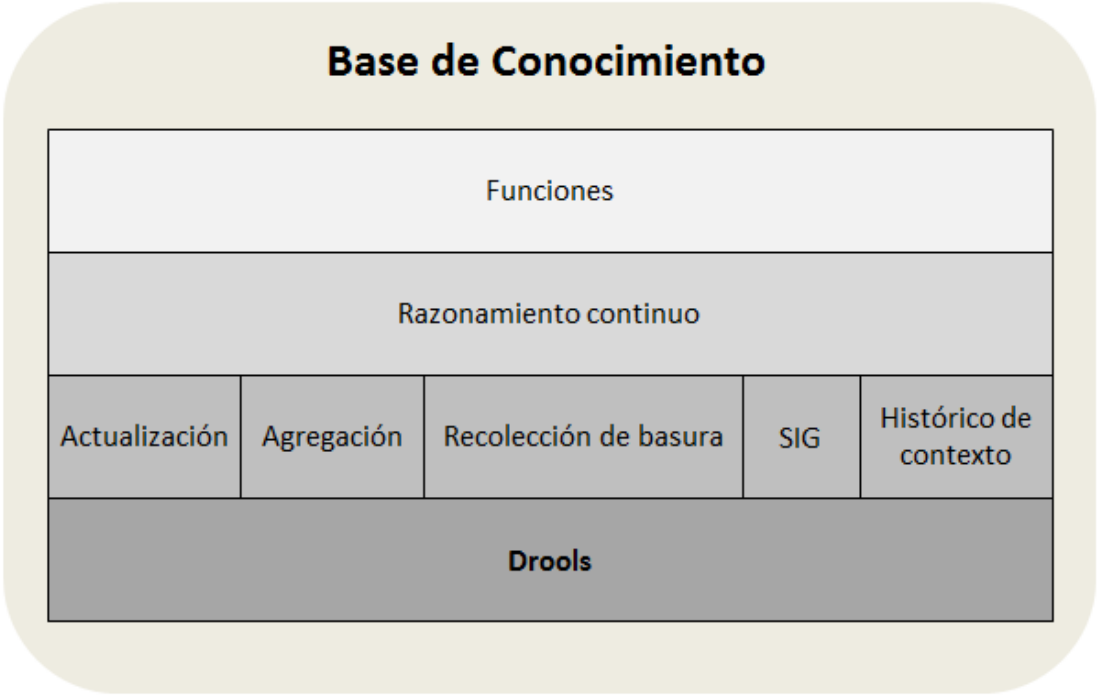


Fig. 63. Arquitectura de la Base de Conocimiento

[43] La elección de *Drools* (http://www.jboss.org/drools/drools-expert.html Último acceso 11-04-2012) viene determinada por el carácter de código abierto del mismo, su implementación en código Java, el cumplimiento de la especificación JSR 94 (http://www.jcp.org/en/jsr/detail?id=94 Último acceso 22-08-2012) por el que ofrece un API estándar para la comunicación con el motor de reglas, la facilidad de extensión del mismo y las funcionalidades soportadas. Existen otros motores de reglas que podrían haber sido utilizados igualmente para la implementación del sistema de reglas de la plataforma (http://java-source.net/open-source/rule-engines Último acceso 22-08-2012), ya que la plataforma ofrece cuadros de diálogo para la configuración de las reglas (Sección 5.8.2), proporcionando mecanismos de abstracción sobre la sintaxis o código de programación específico para cada motor de reglas disponible.

La capa de nivel inferior está compuesta a su vez por diferentes bloques en los que se han implementado aquellas funcionalidades asociadas a la gestión de la información de contexto, las cuales se detallan a continuación.

- *Actualización de la información*: este bloque permite que las entidades definidas en el modelo de datos, cuyas instancias estén presentes en la Base de Conocimiento, puedan ser actualizadas con los nuevos datos provenientes de las fuentes de contexto de manera automática. Así, cada vez que se recibe un nuevo dato relativo a una instancia determinada, el sistema comprobará si el nuevo valor es diferente al que se encontraba almacenado y lo actualizará si procede.

- *Agregación de la información de contexto*: este bloque facilita la agregación de la información de contexto. Así, si existen fuentes y proveedores de contexto que proporcionan diferentes datos para propiedades distintas de una misma instancia de entidad, el sistema realizará la agregación y unificación de dichos datos sobre la instancia de manera automática en base al identificador de la misma.

- *Recolección de basura*: cada vez que una instancia de una entidad de contexto es actualizada en la Base de Conocimiento, se actualiza también una propiedad definida por defecto en todas las entidades, que contiene la fecha y hora de la última actualización. De esta manera, el sistema de recolección de basura es capaz de borrar de la Base de Conocimiento todas aquellas instancias que no hayan sido actualizadas en la última media hora[44]. Esta recolección de basura es importante ya que proporciona un mejor rendimiento para el sistema de reglas, eliminando instancias que no son relevantes para el mismo y propiciando que las reglas tengan que ser evaluadas sobre un menor número de instancias[45].

- *Integración con el sistema SIG*: este bloque integra el motor de reglas con las funcionalidades del sistema SIG, y más concretamente, con la funcionalidad de determinar si una entidad está presente en un área definida. Así, se ha

[44] El periodo de recolección de basura es configurable en la plataforma, posibilitando su desactivación.
[45] En la Sección 6.1 se muestra una evaluación del rendimiento del motor de reglas en la que se evidencia que se mejoran los tiempos empleados en el razonamiento a medida que existen un menor número de instancias almacenadas en la Base de Conocimiento.

añadido al motor de reglas la capacidad de lanzar las reglas asociadas a un área concreta, siempre y cuando la entidad relevante para la situación a detectar se encuentre en dicha área. Así, y teniendo en consideración el caso de uso definido para el presente trabajo, solamente se lanzarán las reglas definidas para detectar la situación "visitante de origen alemán de ruta por la ciudad de Donostia-San Sebastián en un día soleado y a la hora de comer" cuando la persona o visitante se encuentre en el área geográfica delimitada por la ciudad de Donostia-San Sebastián.

- *Gestión del histórico de contexto*: este bloque es el encargado de gestionar el histórico de contexto. Además de poder almacenar de manera manual datos de contexto significativos para las instancias almacenadas en la Base de Conocimiento, el sistema almacena un histórico de las áreas en las que se localizó a una determinada entidad. Así, se pueden conocer las últimas diez áreas y tipos de área en las que una entidad se encontraba. Este histórico puede ser interesante para tener una traza de los lugares en los que una determinada entidad se encontraba ubicada con anterioridad. Un ejemplo de utilización del histórico de contexto puede venir dado por una aplicación orientada al sector turístico, donde se podría saber si una persona ha estado anteriormente en áreas relativas a museos, recreativas, etc. y utilizar ese conocimiento para detectar situaciones, como por ejemplo, "visitante de ruta cultural", "visitante de ocio", etc.

En la siguiente capa se encuentra una funcionalidad implementada para desencadenar un razonamiento continuo sobre las instancias y reglas almacenadas en la Base de Conocimiento.

- *Razonamiento continuo*: el razonamiento sobre las instancias de contexto que se encuentran en la Base de Conocimiento se realiza de una manera periódica y continuada. Así, cada segundo se lanza el mecanismo de razonamiento que evalúa las reglas sobre todas las instancias almacenadas en memoria. Si un ciclo de razonamiento tarda más de un segundo en terminar, el siguiente ciclo de razonamiento esperará hasta que el ciclo en curso acabe. De esta manera, continuamente se están evaluando las reglas sobre la Base de Conocimiento con el fin de tener en cuenta los nuevos datos de contexto recibidos y procesados por la plataforma en tiempo real.

Finalmente, se encuentra una capa en la que se han implementado funciones adicionales a las ya ofrecidas por el motor de reglas con el fin de realizar operaciones sobre los datos de contexto procesados en las reglas, tanto en la parte donde se definen las condiciones de las reglas como en la parte consecuente de las mismas. Las funciones más importantes son las que se explican a continuación.

- *log*: esta función puede establecerse en la parte consecuente de la regla, es decir, en las acciones a efectuar cuando se lanza una regla. Esta función sirve para mostrar un mensaje en la consola de depuración. De esta manera se pueden probar y depurar las reglas que se vayan configurando en la plataforma.

- *POST*: esta función se utiliza para generar las salidas de alto nivel que las reglas pueden producir, por lo que se utiliza en la parte consecuente de las mismas. Esta función puede enviar información asociada a la situación detectada a un punto de acceso web utilizando peticiones HTTP de tipo GET. Para ello, se tiene que especificar la URL del punto de acceso. La información que puede enviarse es toda aquella que esté involucrada en la propia regla, por lo que pueden enviarse datos relevantes que se han utilizado para detectar la situación. En secciones posteriores se muestra la utilización de esta función.

- *rangoTiempo (time*Range)[46]: esta función se puede establecer en la condición de la regla y sirve para determinar el rango horario en el que debe lanzarse la misma. De esta manera se puede especificar la componente temporal asociada a una situación, tal y como se recoge en la definición propuesta en el presente trabajo de investigación.

- *rangoFecha (date*Range): con esta función se establece un rango de fechas en el que se debe ejecutar la regla, por lo tanto puede ser utilizada para establecer las condiciones de la misma. Esta función sirve también para especificar la temporalidad de una situación determinada.

[46] Se da por supuesto que el servidor donde esté alojada la plataforma y el sistema sensible al contexto final comparten la misma zona horaria. La plataforma puede ser configurada para tener en cuenta cualquier zona horaria, utilizando como referencia el tiempo universal coordinado (*Universal Time Coordinated* - UTC)

5.8.2. Controles para la creación de reglas

En esta sección se van a describir los diferentes controles existentes en el diálogo para la creación de reglas proporcionado por la plataforma de desarrollo, tomando como ejemplo el caso de uso planteado.

Como se ha comentado con anterioridad, las reglas que se crean tienen que estar asociadas a un área determinada. Por lo tanto, la regla que tiene que ser creada para detectar la situación del caso de uso de ejemplo tiene que estar asociada al área creada anteriormente de nombre Donostia-San Sebastián.

Para crear la regla, se tiene que seleccionar dicha área y utilizar el control  del cuadro de control de reglas, donde aparecerá el diálogo para la creación de reglas. Este cuadro de diálogo se divide en dos partes. En la parte izquierda del cuadro se encuentran los controles para especificar los datos generales de la regla, así como los cuadros de texto donde se irá generando el código de la regla. En la parte derecha del cuadro, figuran los controles para configurar las condiciones *(when)* y acciones *(then)* de la regla, los cuales generarán el código de la misma de manera automática. El cuadro de diálogo para la creación de reglas se muestra a continuación.

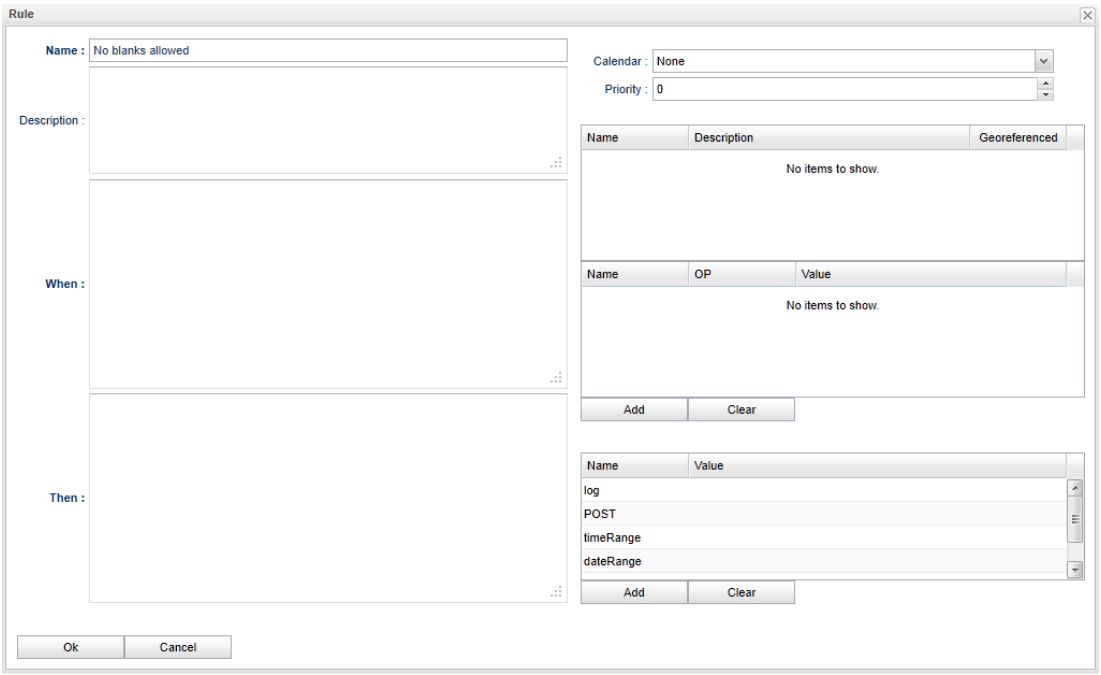


Fig. 64. Diálogo de creación de reglas

De esta manera, los campos que se tienen que proporcionar para la creación de una nueva regla son los que se detallan a continuación.

- *Nombre*: como nombre de la regla puede especificarse el nombre de la situación que se quiere configurar.
- *Descripción*: en el cuadro se puede establecer un texto descriptivo de la regla. Aquí puede indicarse la descripción de la situación proporcionada en la fase de análisis.
- *Cuando (When):* en este recuadro se configuran las condiciones que se tienen que cumplir para que el consecuente de la regla se ejecute. Estas configuraciones pueden realizarse de manera programática mediante la sintaxis de codificación de reglas, o pueden utilizarse los controles que hay en la parte derecha del cuadro de diálogo para introducir las condiciones de manera más sencilla sin tener que codificar la regla manualmente. Estos controles generarán el código de la regla de manera automática.
- *Entonces (Then):* en este recuadro se configura el consecuente de la regla, es decir, las acciones que se llevan a cabo cuando las condiciones definidas se cumplen. Aquí también pueden editarse a mano las consecuencias de la regla o utilizar algunos controles de la parte derecha del cuadro de diálogo para introducir las acciones a realizar de manera no programática utilizando las funciones predefinidas para ello.

En el Anexo II puede encontrarse la descripción de la sintaxis que puede ser utilizada para la creación de las reglas, especificando el antecedente *(when)* y consecuente *(then)* de las mismas de manera programática.

5.8.3. Creación de reglas mediante controles

A continuación se explican los controles disponibles para crear las configuraciones de una regla. Para ello, se tendrán en consideración las entradas y salidas especificadas en la fase de análisis de la situación de ejemplo propuesta, así como las diferentes condiciones especificadas sobre los datos de contexto de entrada.

Así, en primera instancia tienen que ser especificados el nombre de la regla, que será el propio nombre de la situación a ser configurada, y la descripción de la misma, tal y como se refleja en la fase de análisis de la situación. A continuación se muestran ambos campos.

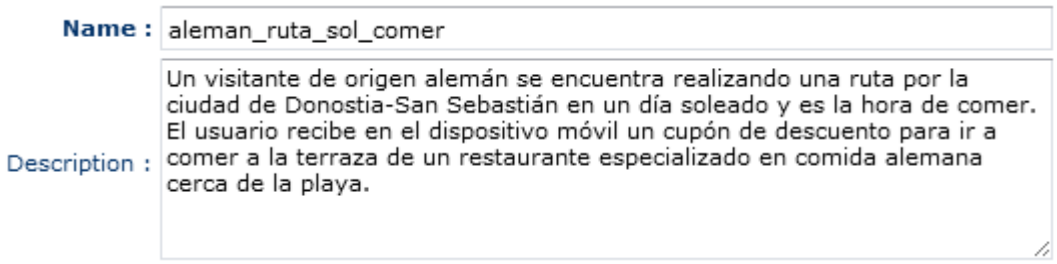


Fig. 65. Especificación de nombre y descripción de la regla

Empezando con la descripción de los controles de la parte derecha que posibilitan la configuración de la regla, se encuentra en primer lugar el control relativo al calendario. Es posible especificar si una regla se tiene que tener en cuenta un día entre semana *(weekday)*, un fin de semana *(weekend)* o un día concreto estableciendo la fecha del mismo. En el caso de uso de ejemplo se va a contemplar cualquier día de la semana, por lo que se dejará el valor a "Ninguno" *("None")*.

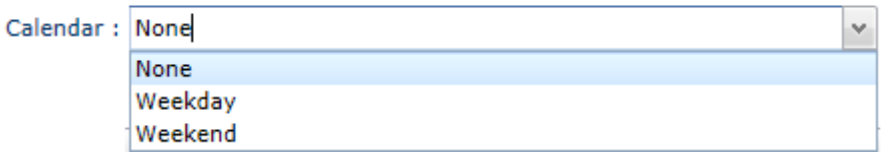


Fig. 66. Calendario de la regla

Además se pueden establecer prioridades en las reglas. A mayor número, mayor prioridad a la hora de procesar la regla por parte del motor de reglas de la plataforma. En este caso concreto la prioridad se establece en 0.

Fig. 67. Prioridad de regla

Los controles que figuran a continuación son los que se utilizan para configurar las condiciones que deben cumplirse sobre los valores que almacenan las propiedades de las instancias de entidades almacenadas en la Base de Conocimiento. Como puede verse en la Figura 68, se muestran las entidades de contexto que están desplegadas en la nube y que pueden ser utilizadas para configurar las condiciones de la regla especificadas en la fase de análisis.

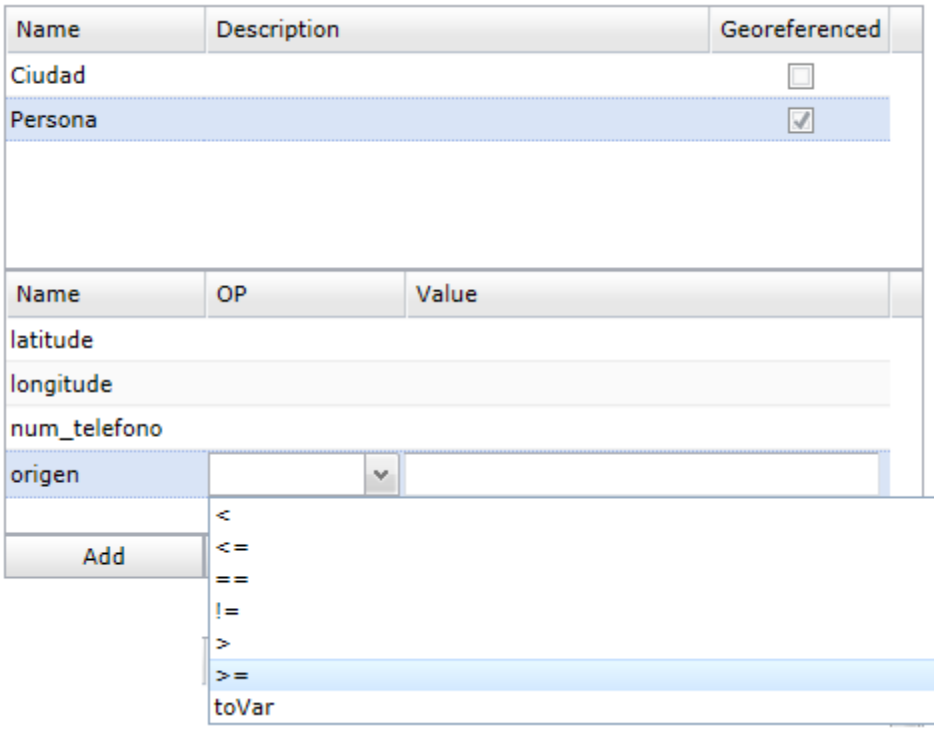


Fig. 68. Configuración de condiciones de la regla

Cuando se elige una entidad de la lista, aparecen además sus propiedades, con el fin de configurar las condiciones en base a valores concretos almacenados en las propiedades. A cada propiedad se le puede asignar un operador y un valor. Los operadores disponibles son comparativos (<, >, >=, <=, !=, ==) y de asignación del valor de la propiedad a una variable mediante la función *"toVar"*. Para añadir una nueva condición sobre la entidad y propiedades seleccionadas se tiene que pulsar el botón "Añadir" *("Add")*.

Para el caso de uso propuesto, se tienen que especificar condiciones sobre las entidades "Persona" y "Ciudad", ya que son las entidades que intervienen de manera directa o indirecta en la situación a detectar. Por una parte se tiene que configurar una condición sobre la entidad "Ciudad", que establezca que la propiedad "temperatura" debe tener un valor superior o igual a 25, especificado en unidades Celsius, y el "nombre" de la ciudad tiene que ser "Donostia-San Sebastián", ya que es el valor que proviene de la fuente de contexto que proporciona los datos meteorológicos para dicha ciudad.

Para especificar ambas condiciones sobre la entidad "Ciudad" se tienen que seleccionar las propiedades, elegir el comparador adecuado y establecer el valor de la

propiedad en las casillas correspondientes, tal y como se muestra en la figura que sigue a continuación.

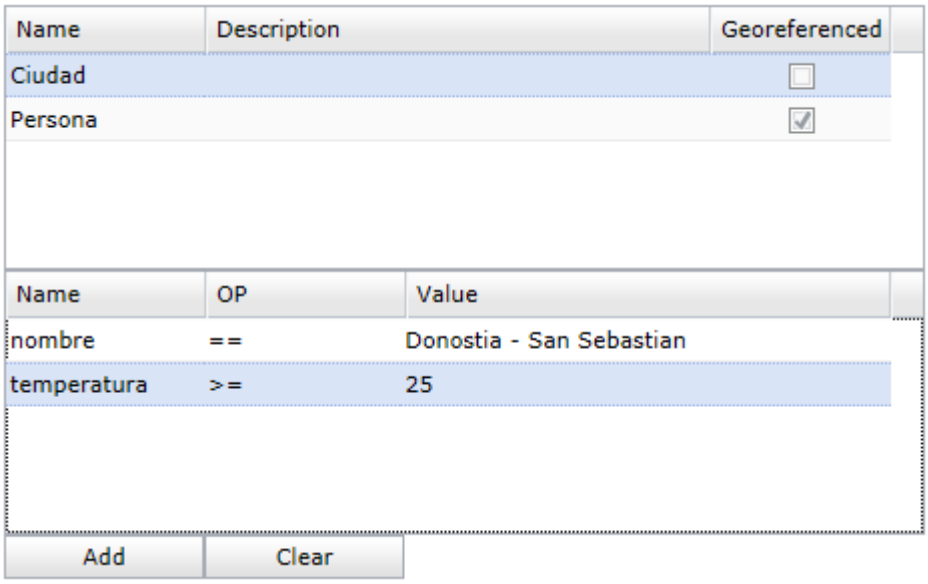


Fig. 69. Configuración de condiciones sobre la entidad "Ciudad"

Una vez que se configura y se añade la condición anterior, se genera el código que se muestra en la Figura 70. Se puede apreciar que se han establecido las condiciones en las propiedades de la entidad indicadas junto con los valores proporcionados.

Además, se asigna la entidad a una variable de manera automática, con el fin de que pueda ser utilizada en cualquier otro punto de la regla. Este código generado puede ser editado a mano en cualquier momento.

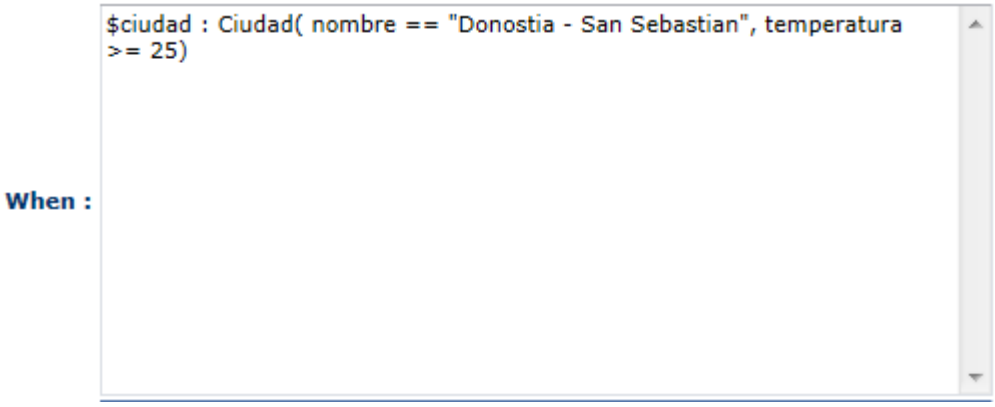


Fig. 70. Código de regla generado mediante controles

Además, se tienen que configurar las condiciones sobre las propiedades de la entidad "Persona". Estas condiciones tienen que establecer que el origen de la persona tiene que ser "Alemania", ya que la guía móvil enviará el origen configurado por el

visitante en su perfil eligiendo el país de procedencia. Esta configuración de la regla se muestra en la siguiente figura.

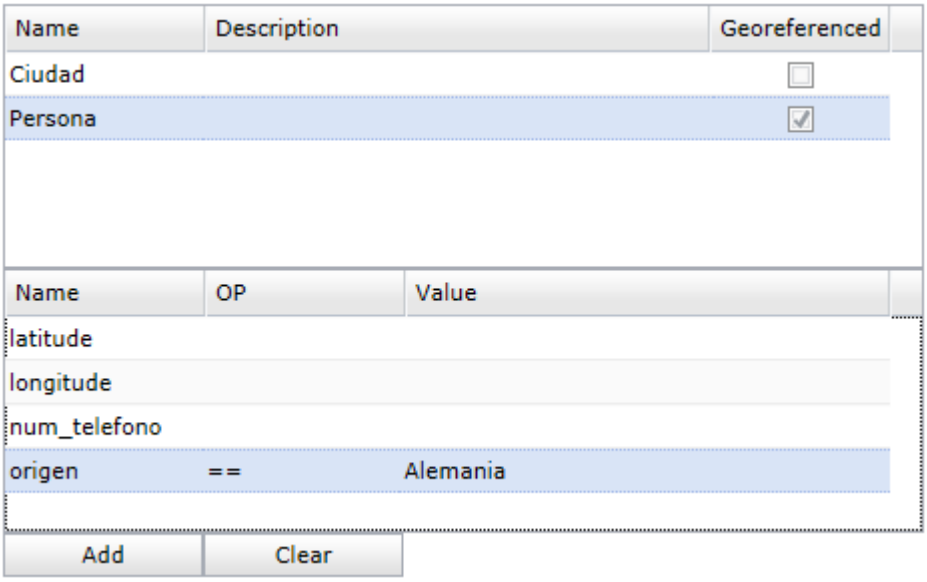


Fig. 71. Configuración de condiciones sobre la entidad "Persona"

Un detalle a tener en cuenta una vez configurada esta condición, es que a la hora de añadir la sintaxis necesaria en el cuadro de condiciones, la plataforma añade una condición extra de manera automática, teniendo así dos condiciones referentes a la entidad "Persona", tal y como se muestra en la siguiente figura.

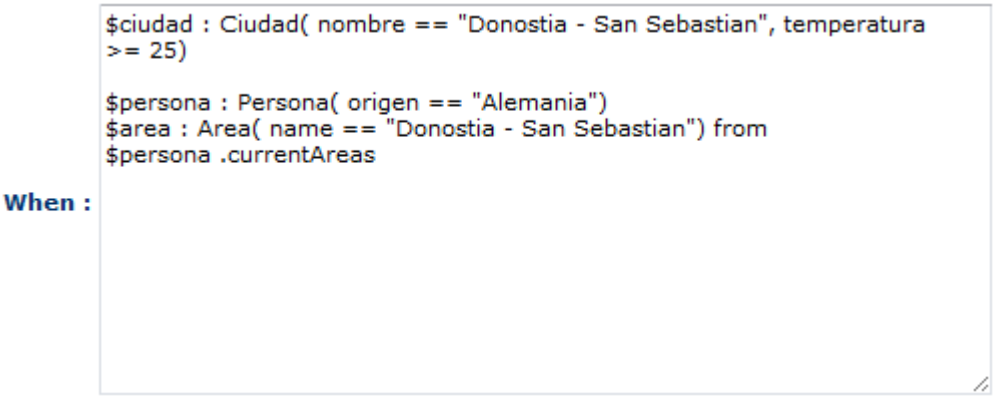


Fig. 72. Condiciones de la regla (a)

La primera condición es aquella en la que aparece la configuración creada para filtrar las personas por origen. La segunda condición sirve para indicar que la entidad, en este caso la "Persona", tiene que estar en el área anteriormente creada con el nombre "Donostia–San Sebastián". Esta condición para restringir la localización de la entidad sobre el área en la que se define la regla se añade siempre que la entidad sobre la que se especifican las condiciones sea de tipo geo-referenciada.

Para realizar el control de las áreas en las que se encuentra una determinada entidad geo-referenciada, el sistema añade por defecto una propiedad a la entidad llamada *"currentAreas"* en la que se guardan todas las áreas en las que la entidad está en un determinado momento, en función de sus parámetros latitud y longitud. El sistema SIG será el encargado de completar la lista de áreas sobre dicha propiedad en función de los parámetros de localización de la entidad que vayan llegando a la plataforma y en función de las áreas definidas en la misma.

Así mismo, toda entidad geo-referenciada dispone también de una propiedad llamada *"pastAreas",* donde se almacenan las diez últimas áreas en las que ha estado dicha entidad, teniendo así un *histórico de contexto* referente a las localizaciones anteriores en las que se ha detectado la entidad.

Para completar la configuración de las condiciones sobre la entidad "Persona", se tiene que configurar la dimensión temporal de la situación con el fin de establecer el rango de tiempo en el que la regla puede ser ejecutada, que en este caso es la hora de comer. Para ello, se tiene que utilizar la función *"timeRange"*, en la que se especifica un rango de tiempo especificado mediante dos horas separadas por comas.

Para el presente caso de uso, se ha tomado como referencia la hora a la que los visitantes de origen alemán suelen comer. La condición especificada mediante el control se muestra en la siguiente figura.

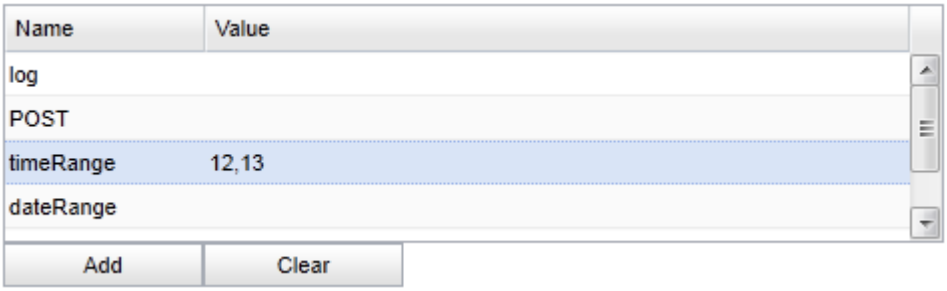


Fig. 73. Condiciones de la regla (b)

La configuración anterior genera de forma automática el código necesario para plasmar dicha condición, tal y como se muestra en la siguiente figura, terminando así con todas las condiciones necesarias para detectar la situación planteada en el caso de uso de ejemplo.

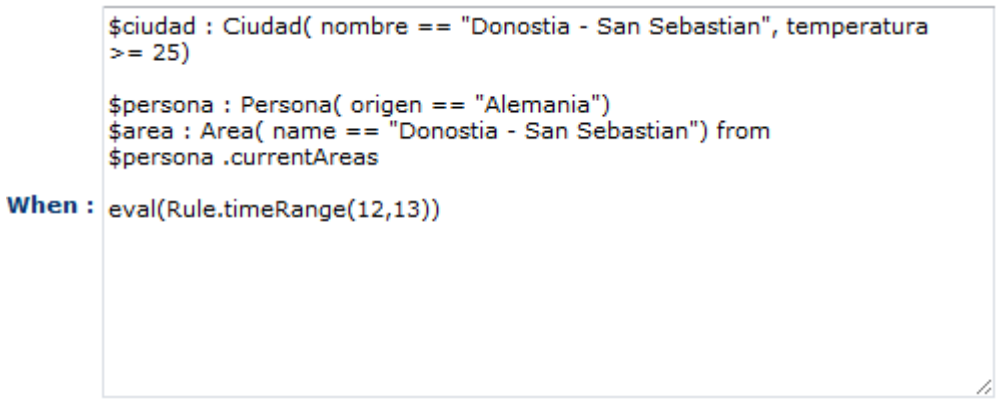


Fig. 74. Condiciones de la regla (c)

El siguiente paso es la configuración de la parte consecuente de la regla, donde se definirán las acciones a realizar una vez que se cumplan las condiciones especificadas anteriormente.

Un primer paso a la hora de configurar las acciones de una regla es la validación de las condiciones definidas con el fin de probar su correcto funcionamiento. Para realizar tales pruebas, se puede utilizar la consola de depuración ofrecida por la plataforma. Para ello, se tiene que especificar en la consecuencia de la regla, un envío de texto a la consola mediante la función *log* que aparece en el cuadro de funciones. De esta manera, para poder validar las condiciones establecidas, se especificará un texto en la función para que sea mostrado en la consola de depuración una vez que se cumplan tales condiciones.

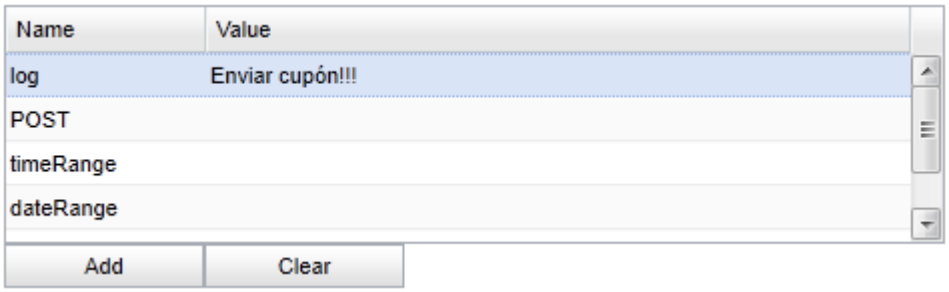


Fig. 75. Configuración del consecuente de la regla (a)

Esta configuración, una vez que se pulse el botón "Añadir" *("Add")*, insertará en el consecuente de la regla la instrucción necesaria para enviar información a la consola de depuración, tal y como refleja la siguiente figura.

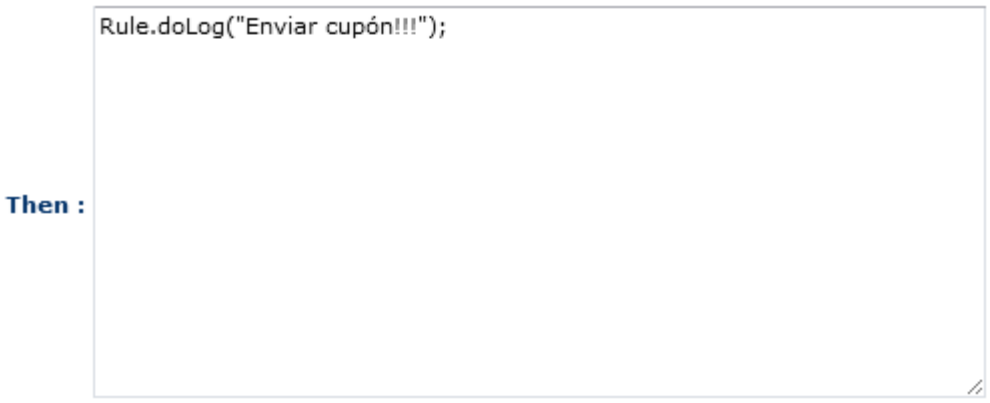


Fig. 76. Configuración del consecuente de la regla (b)

Para terminar la creación de la regla se tiene que pulsar el botón "Ok". Cuando se pulsa dicho botón, la plataforma realiza un proceso de validación de la sintaxis de la regla y avisa al usuario en caso de que exista algún error mediante la consola de depuración.

Una vez configurada la regla, la plataforma está lista para poder recibir y obtener información de contexto de las diferentes fuentes de datos identificadas, desencadenando así el ciclo de vida de la gestión de dicha información en base a todas las configuraciones realizadas con anterioridad.

Así, si se reciben datos referentes a la entidad "Persona" y la entidad "Ciudad" cuyos valores cumplen las condiciones especificadas en la regla creada, la consola de depuración mostrará el mensaje de texto configurado, tal y como se muestra en la figura a continuación.

Fig. 77. Consola de depuración

Además, la función *log* utilizada para mostrar el mensaje por pantalla admite la inclusión de tantas variables de entidades utilizadas en el antecedente de la regla como sean necesarias, separadas por comas. De esta manera, se pueden visualizar también los datos almacenados por las entidades de contexto.

Una vez creada la regla, aparecerá en el listado de reglas, asociada al área sobre la que se llevó a cabo la creación de la misma.

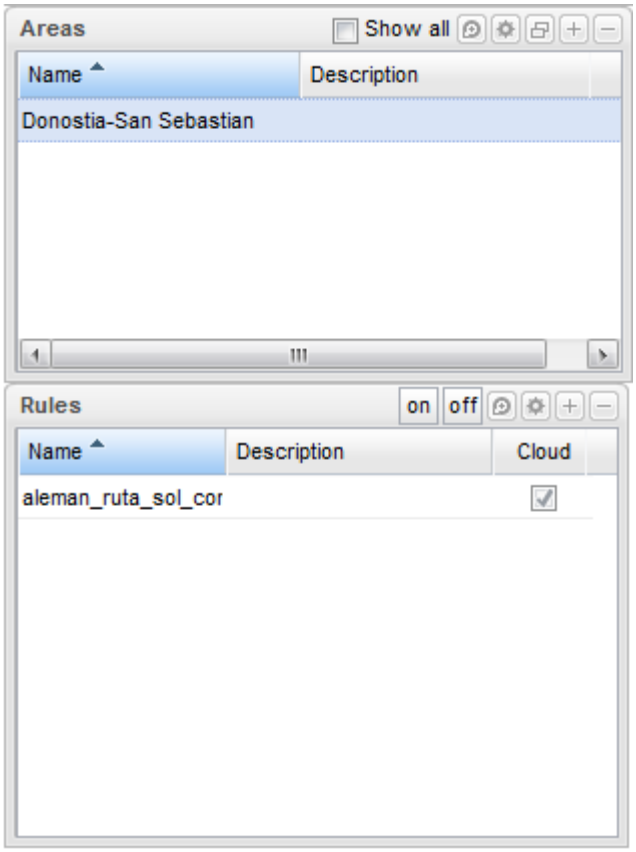


Fig. 78. Listado de reglas creadas

Este cuadro de gestión en el que se listan las reglas creadas, permite además diferentes operaciones sobre las mismas.

- *Visualizar* : visualiza la configuración de la regla seleccionada.
- *Modificar* : permite la modificación de la configuración de una regla.
- *Crear* **:** permite la creación de una regla.
- *Borrar* **:** borra la regla seleccionada.
- *Activar/Desactivar* on off: posibilita la activación o desactivación de una regla determinada.

Además, existe en el listado de reglas un control que indica si la regla está activa en la plataforma *(Cloud)*. Una regla puede ser desactivada de manera intencionada, mediante el control on off, o bien puede ser desactivada por la plataforma en caso de que hubiera algún error en la misma. Un ejemplo de desactivación de reglas por parte de la plataforma podría suceder cuando una de las entidades utilizadas por alguna de las reglas creadas es eliminada. En ese caso, la plataforma desactivaría todas aquellas reglas que

utilizaban la entidad del modelo de datos eliminada, avisando al usuario de la plataforma de dicha acción mediante la consola de depuración.

5.8.4. Generación de salidas de alto nivel mediante reglas

Una vez que se ha comprobado que la regla se lanza correctamente con las condiciones configuradas, el siguiente paso es realizar el envío de información a un punto de acceso o servicio web externo, es decir, generar las salidas con información de alto nivel para que el sistema final pueda utilizarlas con el fin de adaptar su comportamiento. Estas salidas se configuran en función de las salidas especificadas en la fase de análisis. En este caso, las salidas son el nombre de la situación, el identificador de la persona que se ha detectado en dicha situación, que en este caso es el número de teléfono, y la ubicación en la que se detectó la situación de la persona.

En este proceso de configuración de las salidas de la regla tiene más peso el programador ya que utilizará estas salidas para modificar el comportamiento del sistema sensible al contexto en desarrollo. En este caso, se van a utilizar las salidas planteadas para que el servicio de mensajería pueda enviar al dispositivo móvil un cupón de descuento para ir a comer a la terraza de un restaurante cercano.

De esta manera, en vez de enviar el mensaje a la consola de depuración, se enviarán datos de contexto con información sobre la situación detectada al servicio de mensajería, el cual dispone de un servicio configurado para que acepte peticiones HTTP de tipo POST. Con los datos de contexto recibidos, el servicio de mensajería procederá a realizar el envío del mensaje al dispositivo móvil concreto del visitante que se encuentra en la situación detectada.

Para configurar el envío, se tiene que utilizar la función POST, en la que se especifica la URL del punto de acceso remoto, tal y como se muestra en la siguiente figura.

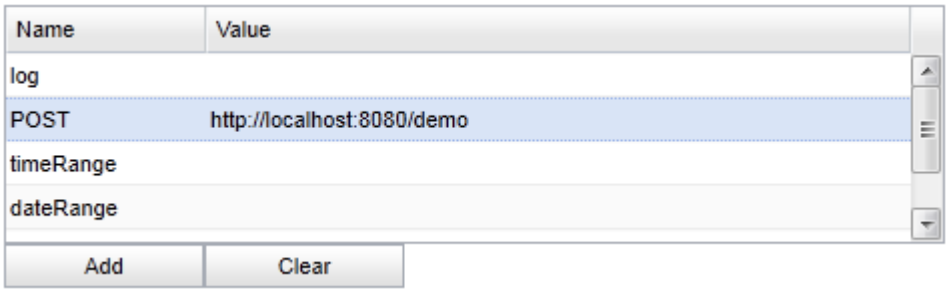


Fig. 79. Función POST

Esta función, añade en el consecuente de la regla la instrucción para enviar datos de contexto a la URL especificada. Además, pueden añadirse tantas entidades o propiedades de estas entidades como se quieran enviar, separadas por comas.

En este caso, se especificarán en la función la totalidad de los datos referentes a la instancia de la entidad "Persona", la cual incluye el número de teléfono y la localización de la persona. Además, y con el fin de ilustrar de mejor manera la funcionalidad POST, se incluye la propiedad "nombre" de la instancia de entidad "Ciudad". A este conjunto de datos se le añade el nombre de la regla de manera automática, el cual sirve como identificador de la situación detectada.

Estos datos de salida se configuran en base a las variables utilizadas en la propia regla y que almacenan las instancias de "Persona" y "Ciudad" que cumplen las condiciones especificadas.

En la siguiente figura se muestra el código generado con la configuración de controles mostrada anteriormente, a la que se le han añadido la variable que almacena la instancia de la entidad "Persona" y el nombre de la instancia de la entidad "Ciudad", el cual es accedido por medio de la variable "$ciudad".

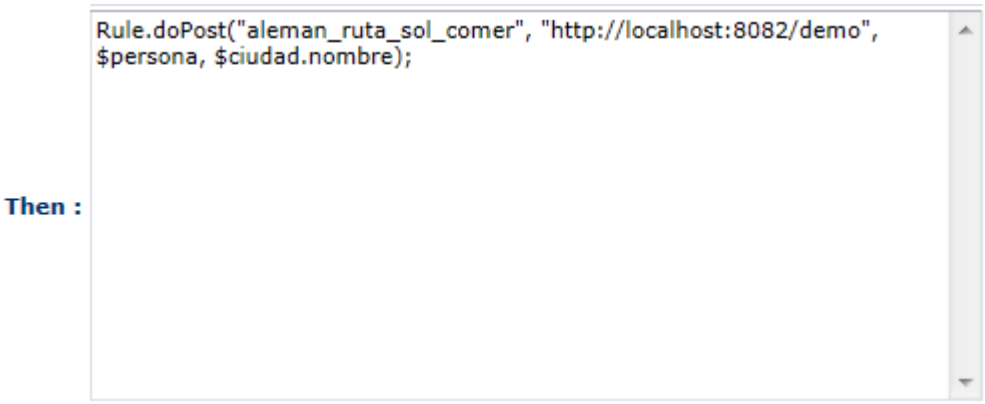


Fig. 80. Función POST configurada

Una vez que la regla se ejecute y se realice la petición POST, se informará por consola de este hecho para comprobar que se ha realizado el envío de manera correcta al servidor remoto especificado.

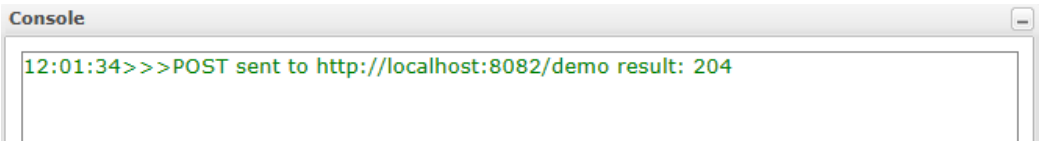


Fig. 81. Consola de depuración función POST

Los datos de salida se envían al servidor remoto en formato XML. En la siguiente figura se muestra el documento XML que es recibido por el servidor en el caso de ejemplo expuesto anteriormente.

```
<rule name="aleman_ruta_sol_comer">
  <entity class="Persona">
    <currentAreas>
      <org.tourgune.cloud.bean.Area>
        <id>10991</id>
        <name>Donostia-San Sebastian</name>
        <type>Ciudad</type>
        <description>Area que identifica el perimetro de la ciudad</description>
        <timestamp>
          <time>1322823673122</time>
          <timezone>Europe/Paris</timezone>
        </timestamp>
      </org.tourgune.cloud.bean.Area>
    </currentAreas>
    <pastAreas>
      <org.tourgune.cloud.bean.Area>
        <id>10991</id>
        <name>Donostia - San Sebastian</name>
        <type>Ciudad</type>
        <description>Area que identifica el perimetro de la ciudad</description>
        <timestamp>
          <time>1322821619991</time>
          <timezone>Europe/Paris</timezone>
        </timestamp>
      </org.tourgune.cloud.bean.Area>
    </pastAreas>
    <updated>
      <time>1322823673122</time>
      <timezone>Europe/Paris</timezone>
    </updated>
    <latitude>43.32183708217748</latitude>
    <longitude>-1.9987821578979492</longitude>
    <num__telefono>605000000</num__telefono>
    <origen>Alemania</origen>
  </entity>
  <entity class="string">Donostia-San Sebastian</entity>
</rule>
```

Fig. 82. XML recibido en el servicio externo

Como se puede observar, el nombre de la regla identifica la situación detectada. De esta manera, el servicio de mensajería podrá configurar los mensajes en relación a la situación concreta detectada. Además, en el documento XML se incluyen todas las propiedades de la entidad "Persona", así como el nombre de la entidad "Ciudad". Esta información puede ser utilizada para que los programadores puedan conocer la situación detectada y tengan la información suficiente para adaptar el sistema final. En este caso, el número de teléfono almacenado en la propiedad "num_telefono" se utiliza para realizar el envío del cupón de descuento para comer en un restaurante cercano. Se utilizan también el nombre de la ciudad y las coordenadas latitud y longitud en las que se

encuentra la persona para recomendar un restaurante concreto que se encuentre cerca del visitante en el momento en el que se detecta la situación.

Cabe destacar que el documento XML, al contener todas las propiedades de la instancia de entidad "Persona", contiene también todas aquellas propiedades que el sistema incluye por defecto y que se han explicado anteriormente, como son las propiedades *"currentAreas"*, donde se muestran los datos de las áreas en las que se detectó a la persona en el momento en el que se identificó la situación, *"pastAreas"*, que guarda el histórico de las áreas anteriores en las que estaba la entidad y la propiedad *"updated"* que indica la fecha y hora de la última actualización de dicha instancia de entidad.

Por otra parte, las salidas de alto nivel pueden también generar nuevas instancias de entidades que sirvan como entradas para la plataforma, tal y como se recoge en la Sección 3.3.1 – RP1.4. Esta funcionalidad posibilita el encadenamiento de reglas en base a las salidas generadas por las mismas.

5.9. Resumen

En este capítulo se ha detallado tanto la arquitectura como las funcionalidades principales de cada uno de los módulos que conforman la plataforma *Context Cloud*. La implementación de la plataforma se ha realizado teniendo en cuenta tanto el marco teórico establecido como los requerimientos identificados en base a dicho marco teórico y las carencias detectadas en la revisión del estado del arte en relación a las herramientas para el desarrollo de sistemas sensibles al contexto. Además, se ha tenido en cuenta la metodología diseñada para dar soporte y guiar el ciclo de vida del desarrollo de los sistemas sensibles al contexto.

Como resumen sobre la plataforma implementada, se van a analizar tanto el cumplimiento de los requisitos especificados en la definición del marco teórico, así como una comparativa con el resto de soluciones analizadas en el estado del arte en base a los criterios de evaluación establecidos.

5.9.1. Cumplimiento de los requisitos especificados

En la Sección 3.3 se han especificado ciertos requisitos que este tipo de plataformas para el desarrollo de sistemas sensibles al contexto deben tener con el fin de cubrir las carencias encontradas en el estado del arte. El principal punto de partida ha sido la

definición de los términos *contexto* y *situación*, donde se presta especial atención a los condicionantes temporales y espaciales que determinan una situación.

De esta manera, la implementación de la plataforma ha tratado de dar respuesta a todos y cada uno de los requisitos establecidos. De forma resumida, se hace un repaso a todos los requisitos, detallando el cumplimiento de los mismos en base a las características y funcionalidades implementadas en la plataforma.

5.9.1.1. **Arquitectura de la plataforma**

La arquitectura de la plataforma se basa en los tres requisitos establecidos. Por una parte, se ha centrado en la aplicación de las buenas prácticas establecidas en la comunidad de la Web de las Cosas (RP1.1), posibilitando la interacción de la plataforma con las diferentes fuentes de contexto utilizadas en base a peticiones HTTP. De esta forma se da un mecanismo homogéneo de acceso a cualquier tipo de fuente de contexto, siempre y cuando esté adaptada para poder servir los datos de contexto mediante las peticiones POST o GET (Sección 5.3). Es cierto que se posibilita que la información que llega a la plataforma esté solamente representada en formato XML, pero sería fácilmente ampliable a otros formatos, como por ejemplo JSON.

Además, se ha prestado especial atención a la interacción de los usuarios con la plataforma, ya que estos pueden no tener conocimientos de programación (RP1.2). Por lo tanto, se ha diseñado una interfaz web donde las diferentes configuraciones para gestionar la información de contexto se realiza mediante diálogos (Sección 5.2.1). Además, la plataforma gestiona las configuraciones introducidas y genera el código necesario para poder procesar la información de contexto. De esta manera los usuarios no tienen que implementar una sola línea de código para poder gestionar la información de contexto.

Por su parte, la plataforma puede ser desplegada en cualquier infraestructura de servidores que soporte el despliegue de aplicaciones web, por lo que puede ser gestionada por infraestructuras hardware y software alojadas "en la nube" (RP1.3), simplificando así su mantenimiento y despliegue a los usuarios.

Finalmente, se ha tenido en consideración el carácter reactivo de los sistemas sensibles al contexto (RP1.4), posibilitando que la plataforma produzca salidas con información de alto nivel en base a los datos de bajo nivel procesados (Sección 5.8.4). Estas salidas son las que los sistemas finales utilizarán para reaccionar y adaptar su comportamiento.

5.9.1.2. **Modelo de datos**

El modelo de datos también cumple los diferentes requisitos especificados. Por una parte se han tenido en cuenta los diferentes requisitos de carácter general que el modelo de datos debe cumplir (RP2.1). Así, los diálogos para la creación del modelo de datos permiten definir un modelo de datos adaptable a cualquier tipo de información de contexto proveniente de las fuentes, ya que se permiten especificar propiedades de diferentes tipos de datos (Sección 5.6). Además, se permiten definir dependencias y relaciones entre las diferentes entidades definidas. Estas dependencias pueden configurarse en función de los identificadores especificados en cada una de las entidades, estableciendo posteriormente unificaciones de dichas entidades a nivel de reglas (Anexo II).

El modelo de datos configurado mediante diálogos se traduce de manera automática a código Java que es gestionado por la plataforma, por lo que el modelo de datos es un modelo orientado a objetos. Este modelo es, por lo tanto, compatible con los mecanismos de inferencia utilizados por la plataforma *Drools*.

Finalmente, los mecanismos establecidos para la creación del modelo de datos permiten una gran flexibilidad ya que se pueden crear las entidades y propiedades que se requieran en cada momento.

Además de estos requisitos de carácter general, se ha prestado especial atención a los requisitos espaciales (RP2.2) y temporales (RP2.3) que el modelo de datos debe soportar en relación a la definición de situación establecida en el presente trabajo de investigación. De esta manera, existe la posibilidad de configurar una entidad como geo-referenciada, de manera que la plataforma añada de forma automática las propiedades necesarias para almacenar las coordenadas de dicha entidad que serán gestionadas por el módulo SIG (Sección 5.6). A su vez, se permite la creación de áreas como parte del modelo que gestiona la plataforma, con el fin de establecer las regiones geográficas en las que se puede localizar a las entidades geo-referenciadas (Sección 5.5).

En cuanto al soporte temporal, y como se ha descrito con anterioridad, cada una de las entidades contiene por defecto una propiedad denominada *updated* que es gestionada por la plataforma y que guarda la última vez en la que se actualizó la entidad. Esta propiedad es utilizada por el motor de reglas para realizar comparaciones temporales (Anexo II). Además, existen tipos de datos temporales (fechas) que las propiedades pueden almacenar (Sección 5.6).

5.9.1.3. **Razonamiento**

En cuanto al razonamiento, se ha implementado un sistema de inferencia basado en reglas (RP3.1) que permite al experto en el dominio explicitar su conocimiento en base a dichas reglas con el fin de detectar las situaciones relevantes para el sistema a implementar (Sección 5.8.2).

Además, se han desarrollado sobre el motor de reglas utilizado, algunas capas con funcionalidades adicionales, una de las cuales está orientada a dar soporte espacial al razonamiento (RP3.2) (Sección 5.8.1). Así, se permite establecer como condición la presencia de una determinada entidad en un área concreta creada en la plataforma (Sección 5.8.3).

Finalmente, la propiedad *updated* que contienen todas las entidades por defecto, permite realizar operaciones temporales (RP3.3) en base a la lógica de Allen soportada por el motor de reglas utilizado[47] (Anexo II).

5.9.1.4. **Gestión automática de la información de contexto**

Un requisito fundamental en el funcionamiento de la plataforma es la automatización en la gestión de la información de contexto, con el fin de eliminar el código programático a implementar por parte de los usuarios de la plataforma. Estas funciones de automatización incluyen, por ejemplo, la transformación de la información de contexto al modelo de datos definido de manera automática (R4.1). Estas transformaciones se realizan en base a las configuraciones de mapeo que la plataforma permite realizar mediante diálogos (Sección 5.7). De esta manera, toda la información que llega a la plataforma puede ser modelada para ser utilizada posteriormente en la definición de las reglas.

Además, la inserción y actualización de las instancias almacenadas en la Base de Conocimiento en base al modelo de datos definido se realizan de manera automática (RP4.2). Estas operaciones se realizan en base al identificador que cada una de las instancias almacenadas posee de manera obligatoria. Es este identificador el que también permite que datos provenientes de diferentes fuentes de contexto se agreguen y unifiquen (RP4.3) en una misma instancia de manera automática (Sección 5.7).

Finalmente, también se ha implementado un mecanismo de recolección de basura automática, que elimina de la Base de Conocimiento instancias de entidades

[47] http://docs.jboss.org/drools/release/5.4.0.CR1/drools-fusion-docs/html/ch02.html#d0e547 Último acceso 13-04-2012

pertenecientes al modelo de datos definido que ya no son relevantes para la ejecución de las reglas (RP4.4). La relevancia se ha tenido en cuenta en función de la fecha y hora de la última actualización de la instancia, eliminando así las instancias que no fueron actualizadas en los últimos treinta minutos (configurable) (Sección 5.8.1).

5.9.1.5. **Extensible**

La plataforma es extensible en relación a diferentes aspectos de funcionamiento de la misma. Así, se permite la configuración en tiempo real de nuevos proveedores que accedan a fuentes de contexto adicionales que se requieran para poder detectar nuevas situaciones o identificar con mayor precisión las situaciones ya configuradas (RP5.1). De este modo, no es necesario volver a implementar nada, ya que se realiza la actualización del sistema en tiempo real con el mínimo de impacto sobre el funcionamiento de la plataforma y del sistema final.

Otra de las posibilidades que ofrece la plataforma, es la modificación del modelo de datos en tiempo real (RP5.3). Estas modificaciones del modelo de datos permiten dar respuesta a las nuevas incorporaciones de fuentes de contexto comentadas anteriormente. La plataforma controla en todo momento los cambios realizados sobre el modelo de datos y valida tanto los mapeos configurados como las reglas configuradas en base a dicho modelo, informando al usuario de la plataforma de posibles conflictos como consecuencia de las modificaciones del modelo de datos (Sección 5.6).

Finalmente, las reglas también pueden ser modificadas en tiempo real (RP5.3). Esta modificación de reglas se suma a las anteriormente comentadas, ofreciendo así al usuario la posibilidad de cambiar en caliente la gestión de la información de contexto necesaria para identificar las situaciones relevantes (Sección 5.8.3).

Esta extensibilidad de la plataforma permite incorporar nueva información de contexto para aumentar la precisión en la detección de situaciones, eliminar situaciones que ya no son relevantes para el sistema final o añadir nuevas situaciones de manera sencilla y con el mínimo impacto sobre las configuraciones realizadas en la plataforma anteriormente. Además, muchos de estos cambios tienen también un impacto mínimo en el sistema final que utiliza las salidas generadas, siempre y cuando estas salidas no cambien, sino que cambien las configuraciones que las generan.

5.9.1.6. **Movilidad**

La movilidad es un factor clave tanto en la propia concepción de situación que se realiza en el presente trabajo como en la implementación de la plataforma (RP6). Por lo

tanto, se ha prestado especial atención en dotar a la plataforma de mecanismos que permitan gestionar la movilidad de las entidades de una manera automatizada. Así, el modelo de datos permite la definición de entidades móviles con parámetros relativos a las coordenadas de la misma que son gestionados por el sistema SIG incorporado en la arquitectura de la plataforma (Sección 5.6).

Además, se permite la definición de áreas geográficas en las que poder ubicar a las entidades con movilidad y poder detectar así situaciones relevantes en dichas regiones espaciales (Sección 5.5). Se ha incorporado también la posibilidad de establecer condiciones espaciales en las configuraciones de las reglas (Sección 5.8.3).

5.9.1.7. **Colaboración**

El establecer un proceso de diseño y desarrollo colaborativo para la implementación de sistemas sensibles al contexto entre programadores y expertos en el dominio es un pilar fundamental en el presente trabajo de investigación. La plataforma ha sido diseñada para posibilitar la aplicación de la metodología propuesta para la definición y configuración de situaciones (Capítulo 4) la cual tiene como base la colaboración entre programadores y expertos en el dominio, por lo que la propia plataforma tiene también un carácter colaborativo (RP7.1).

Se ha diseñado la interfaz de la misma con el fin de tener un control sobre todos los parámetros de configuración a simple vista y poder realizar las configuraciones para la gestión de la información de contexto mediante diálogos (RP7.2), de tal manera que tanto programador como experto en el dominio controlen en todo momento los aspectos de configuración y puedan ser partícipes de dicha configuración de una manera sencilla y visual (Sección 5.2.1).

5.9.1.8. **Web**

La plataforma se ha implementado con soporte web, de tal manera que es accesible desde cualquier dispositivo con navegador web y conexión a Internet (RP8). Además, tal como se ha comentado anteriormente, posibilita su despliegue en infraestructuras para la computación en nube (RP1.3) gracias a su naturaleza web. Este hecho facilita el acceso a la plataforma, no teniendo que realizar ningún tipo de instalación y mantenimiento hardware. Además, acelera el proceso de adopción de la misma, pudiendo modificar las configuraciones realizadas desde cualquier lugar y en cualquier momento a través de la interfaz web.

5.9.2. Comparativa con el resto de soluciones analizadas

Tal y como se recoge en el estado del arte, las diferentes soluciones analizadas no disponen de todas las características deseables para poder dar soporte al desarrollo de sistemas sensibles al contexto. Estas características y funcionalidades, se han incorporado a la plataforma implementada con el fin de cubrir las carencias detectadas en las soluciones estudiadas.

La Tabla 7 recoge las características que fueron analizadas en los diferentes entornos de desarrollo, y muestra también las características implementadas en la plataforma *Context Cloud*. Se puede observar que la plataforma implementada cumple con todas las características establecidas como fundamentales para este tipo de entornos de desarrollo.

Todas las funcionalidades que se deben incorporar están alineadas con el marco teórico y los requisitos planteados inicialmente sobre las características de la plataforma. Así, la plataforma se basa en un modelo de datos orientado a objetos. Es extensible tanto en la incorporación de nuevas fuentes, en el modelo de datos como en la configuración de las reglas, dando soporte al dinamismo que este tipo de sistemas tiene. Dispone de una gestión automática de la información de contexto y posee mecanismos de razonamiento para la detección de situaciones y soporte para movilidad.

Un aspecto clave en las características de la plataforma es la orientación hacia usuarios no expertos en programación, que en este caso son los expertos en el dominio, contando así con un entorno web y visual, donde las configuraciones se realizan sin tener que programar ningún tipo de funcionalidad para la gestión de la información de contexto. Además, la plataforma puede ser desplegada en una infraestructura en la nube, abaratando costes de mantenimiento y acelerando la adopción de la misma.

De esta manera, la plataforma implementada responde a las necesidades y carencias detectadas en el estado del arte, contribuyendo a una mejora tanto del proceso de implementación de sistemas sensibles al contexto, guiada por la metodología de desarrollo propuesta, como del sistema final a implementar.

Tabla 7. Comparativa de la plataforma *Context Cloud* con el resto de soluciones analizadas

	Modelo	**Extensible**	**Gestión automática**	**Razonamiento**	**Movilidad**	**Usuario no técnico**	**Entorno Visual**	**Web**
Context Toolkit	Pares clave-valor	No	No	Sí	No	No	No	No
SOCAM	Ontologías	No	No	Sí	No	No	No	No
CoBra	Ontologías	No	No	Sí	No	No	No	No
JCAF	Orientado a Objetos	No	No	No	No	No	No	No
Semantic Space Toolkit	Ontologías	No	No	Sí	No	No	No	No
CASS	Relacional	No	No	Sí	No	No	No	No
CMF	Ontologías	No	No	Sí	Sí	No	No	No
Hydrogen	Orientado a Objetos	No	No	No	Sí	No	No	No
DiaSuite	Orientado a Objetos	No	No	Sí	No	Sí	Sí	No
OPEN	Ontologías	No	No	Sí	No	Sí	Sí	Sí
Context Cloud	Orientado a Objetos	Sí	Sí	Sí	Sí	Sí	Sí	Sí

Capítulo 6 Evaluación

Este capítulo recoge la evaluación tanto de la metodología para el desarrollo de sistemas sensibles al contexto propuesta en el Capítulo 4 como del uso de la plataforma para el desarrollo de dichos sistemas detallada en el Capítulo 5 y del rendimiento de la misma.

Así, en primera instancia, la evaluación se centra en el análisis del rendimiento ofrecido por el sistema de razonamiento de la plataforma. Este tipo de entornos tienen que hacer frente a la gestión de grandes cantidades de información de contexto en tiempo real, por lo que un posible cuello de botella es el propio proceso de razonamiento. Cuantas más instancias de entidades de contexto se encuentren almacenadas en la Base de Conocimiento, mayores serán los tiempos empleados por el sistema para poder comprobar las reglas configuradas sobre cada una de las instancias almacenadas (Lamsfus et. al., 2011). Por lo tanto, es indispensable realizar una evaluación del rendimiento que la plataforma implementada puede ofrecer con el fin de ver cómo de escalable resulta su arquitectura y qué tiempos se consiguen en los procesos de razonamiento en función de ciertos parámetros como el número de instancias y reglas que tengan que ser evaluadas sobre la Base de Conocimiento.

Por otra parte, resulta imprescindible realizar una evaluación tanto de la metodología como de la plataforma con usuarios reales que puedan contrastar la hipótesis de partida. Así, se pretende validar que con herramientas de desarrollo y metodologías adaptadas es posible involucrar a personal no técnico, como los expertos en el dominio, en el proceso de desarrollo en colaboración con el personal programador, mejorando así el propio desarrollo de sistemas sensibles al contexto y por consiguiente, el sistema final resultante.

La metodología utilizada para la evaluación de la plataforma con usuarios reales se enmarca dentro de las corrientes de conocimiento denominadas como *Human-Computer Interaction (HCI)*, pertenecientes a la disciplina *Management Information Systems (MIS)*. Se trata de una disciplina consolidada en las últimas décadas que ha generado una serie de métodos robustos que permiten valorar el grado de utilidad y usabilidad de diversas soluciones tecnológicas, en particular, sistemas e interfaces que los usuarios deben manejar.

Este campo de conocimiento se centra en entender la forma en la que las personas interactúan con la información, la tecnología y las tareas a ejecutar, especialmente en los contextos empresariales, culturales y organizaciones (Zhang y Li, 2004). Entre los objetivos operativos se busca minimizar errores, disminuir la sensación de frustración y generar condiciones más apropiadas con el fin de conseguir que las tareas que implican la interacción entre persona y ordenador sean más productivas. Por tanto, la investigación busca la estandarización de la usabilidad y la mejora de los desarrollos de herramientas de usuario.

De esta manera, el escenario de evaluación que se plantea en este trabajo de investigación presenta el uso de la metodología y de la plataforma en un entorno laboral. Así, tal y como se describe en la Sección 4.1, el experto en el dominio es considerado como el cliente que requiere los servicios de desarrollo de los programadores con el fin de implementar un sistema sensible al contexto aplicable al dominio concreto de aplicación del que es experto. Es por lo tanto el programador el que dispone de las herramientas de desarrollo necesarias y las metodologías aplicadas para poder involucrar al experto en el dominio en el ciclo de vida del desarrollo del sistema a implementar. De esta manera, el propio experto en el dominio puede estar presente en dicho desarrollo desde sus inicios.

En la evaluación diseñada, se ha establecido como dominio de aplicación el entorno turístico. Se ha simulado un entorno de trabajo donde un cliente (un gestor de destino) tiene la necesidad de implementar un servicio móvil turístico para que los visitantes (usuarios finales) puedan recibir información y servicios adaptados a su contexto.

En siguientes secciones se detallan tanto la evaluación del rendimiento del motor de reglas de la plataforma implementada, así como la evaluación de usuario realizada y los resultados obtenidos.

6.1. Evaluación del rendimiento del motor de reglas

En esta sección se describe la metodología utilizada para llevar a cabo la evaluación del rendimiento de la plataforma en relación al tiempo de razonamiento sobre las instancias almacenadas en la Base de Conocimiento, así como los resultados y conclusiones obtenidas.

El sistema de razonamiento que la plataforma utiliza se fundamenta en el algoritmo RETE (Forgy, C., 1982) el cual permite optimizar el proceso de razonamiento mediante reglas aplicadas sobre una serie de hechos que componen la Base de Conocimiento, que en este caso son las instancias de las entidades que sirven para modelar la información de contexto. Este algoritmo crea una red de nodos basada en las condiciones establecidas en las reglas los cuales se evalúan utilizando cada una de las instancias almacenadas en la Base de Conocimiento[48].

6.1.1. Metodología

Para realizar la evaluación del rendimiento de la plataforma, se han utilizado las funcionalidades expuestas por el módulo donde reside la Base de Conocimiento junto con el motor de reglas. De esta manera, se ha implementado una interfaz gráfica que permite la comunicación con el motor de reglas y posibilita la definición de diferentes variables relevantes para cada una de las pruebas de rendimiento planteadas. La interfaz gráfica se muestra a continuación.

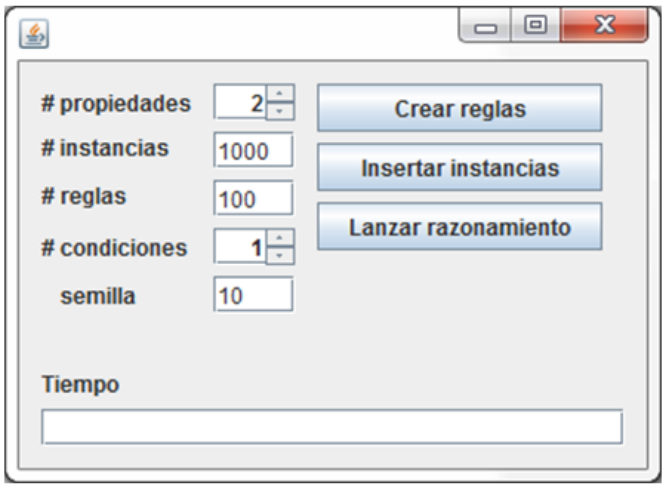


Fig. 83. Interfaz de control para la evaluación del rendimiento del motor de reglas

[48] http://techondec.wordpress.com/2011/03/14/rete-algorithm-demystified-part-2/ Último acceso 08-05-2012.

Existen diferentes campos donde se pueden especificar los parámetros necesarios a tener en cuenta para las pruebas de evaluación. La descripción de dichos parámetros se detalla a continuación.

- # *propiedades*: número de propiedades que tienen los objetos o instancias que se crean e insertan en la Base de Conocimiento. Todas las propiedades creadas son de tipo entero. Además, por defecto, cada instancia dispone de una propiedad que puede almacenar el identificador de la misma, ya que la Base de Conocimiento requiere que cada instancia sea identificada de manera unívoca.

- # *instancias*: número de instancias que se crean e insertan en la Base de Conocimiento. Sobre el número indicado, se crean instancias de dos tipos de entidades diferentes, las cuales se han implementado utilizando dos clases Java de nombres "Bean_A" y "Bean_B". Así, la mitad de las instancias generadas son de un tipo de entidad y la otra mitad, del tipo restante. Además, en el proceso de creación se asignan números enteros de manera aleatoria a las propiedades creadas en cada una de las instancias.

- # *reglas*: número de reglas que se crean y que son insertadas en la Base de Conocimiento para ser evaluadas contra las instancias almacenadas previamente. Las reglas se crean de manera automática en función de las entidades definidas por las clases Java "Bean_A" y "Bean_B", las propiedades que cada una de las instancias contenga y una serie de operadores de comparación (<, >, ==) que se aplican sobre cada una de las propiedades de las entidades utilizadas. Todas las diferentes configuraciones que una regla puede tener, se generan de manera aleatoria.

- # *condiciones*: entidades sobre las que se configuran las condiciones de sus propiedades. El número de condiciones especificado puede ser uno o dos. Si el número especificado es uno, las reglas contendrán condiciones solamente sobre un tipo de entidad ("Bean_A" o "Bean_B"), mientras que si el valor es dos, se crearán condiciones sobre los dos tipos de entidades.

- *Semilla*: número máximo que puede asignarse a las propiedades de las instancias creadas, es decir, que dichas propiedades almacenarán un valor aleatorio entre cero y el número especificado como semilla.

Una vez que se especifican todos los parámetros, se pueden utilizar los controles situados a la derecha del panel para evaluar el rendimiento del sistema de razonamiento. Así, los controles que se proporcionan son los siguientes.

- *Crear reglas*: pulsando este botón, se crean las reglas indicadas en el parámetro "# reglas". Estas reglas contendrán condiciones sobre las entidades "Bean_A" y/o "Bean_B". Las condiciones dependen del número de propiedades especificado en "# propiedades", del valor proporcionado como semilla y del parámetro "# condiciones". Como ejemplo, si se configura que el número de propiedades sea 2, el número de condiciones sea 1 y la semilla sea 10, el sistema podría crear la siguiente regla de manera aleatoria y almacenarla en la Base de Conocimiento.

```
1.  when
2.     $b0 : Bean_B(a==0, b>5)
3.  then
4.     Rule.doPost("rule", "http://my.server:8080", $b0);
5.  end
```

Como puede observarse en la línea 2, se crea la condición sobre las dos propiedades creadas en la entidad "Bean_B". Tanto el tipo de entidad que se establece en la regla, como los operadores que conforman las condiciones sobre las propiedades y los propios valores que las propiedades almacenan se crean de manera aleatoria. Además, en la línea 4 figura la acción a realizar una vez se den las condiciones definidas, donde se llama a la función encargada de preparar el envío de los datos pertenecientes a las instancias mediante una petición de tipo POST al servicio configurado para tal fin (Sección 5.8.5).

Si se especifica que el número de condiciones sea 2, la regla se crea utilizando los dos tipos de entidades sobre los que establecer las condiciones. Una posible regla resultante sería la siguiente.

```
6.  when
7.     $b0 : Bean_A(a==3, b<5 )
8.     $b1 : Bean B(a>4, b>8 )
9.     eval($b0.id < $b1.id);
10. then
11.    Rule.doPost("rule", "http://my.server:8080", $b0);
12. end
```

Así, en las líneas 7 y 8 se especifican las condiciones sobre las propiedades de los dos tipos de entidades que utiliza el sistema de evaluación. Además, en la línea 9 se establece como condición que el identificador de una de las instancias debe ser menor que el identificador de la otra instancia. Esta tercera condición evita que el sistema de razonamiento evalúe más entidades de las que debiera.

- *Insertar instancias*: pulsando este botón se crean todas las instancias especificadas y se insertan en la Base de Conocimiento. Como se ha comentado anteriormente, las instancias pueden ser creadas a partir de dos tipos de entidades, "Bean_A" y "Bean_B". Los valores almacenados en las propiedades se generan de forma aleatoria.

- *Lanzar razonamiento*: este control posibilita el lanzamiento del proceso de razonamiento sobre las instancias almacenadas. El tiempo invertido en dicho razonamiento aparece posteriormente en el campo "Tiempo", especificado en milisegundos.

La máquina sobre la cual se han ejecutado las pruebas es un HP EliteBook 2540p, con sistema operativo Windows 7 y máquina virtual Java versión 1.6. Además, la máquina virtual se ha configurado con una pila de almacenamiento de objetos en memoria de un gigabyte de capacidad.

6.1.2. Resultados

Las pruebas se han realizado utilizando diferentes configuraciones sobre los parámetros que posibilita introducir la interfaz gráfica descrita con anterioridad. Concretamente, se ha variado el número de propiedades de las entidades, el número de instancias, el número de reglas y el número de condiciones establecidas en las diferentes pruebas realizadas. La semilla se ha fijado por defecto a un valor de 10 para todas las pruebas realizadas, por lo que este parámetro no ha sido utilizado como determinante en las pruebas.

Además, por cada una de las configuraciones utilizadas en cada una de las pruebas planteadas, se ha lanzado el proceso de razonamiento tres veces con el fin de obtener una media del tiempo de ejecución empleado. Se considera que un tiempo de razonamiento aceptable debe ser inferior a los 1.000 milisegundos (Miller, 1968), con el fin de que la adaptación del sistema sensible al contexto, y por lo tanto, la

personalización del servicio utilizado por el usuario final sea lo más inmediata posible y no se vea afectada la experiencia de usuario en la utilización del servicio.

A continuación se detallan las pruebas de evaluación de rendimiento (PER#) que se han llevado a cabo, las configuraciones utilizadas en cada una de las mismas y los resultados obtenidos.

6.1.2.1. **PER1 - Número de propiedades**

La primera prueba se ha realizado con el fin de saber cómo afecta el número de propiedades que disponen las instancias de las entidades utilizadas sobre las que se configuran las condiciones de las reglas.

Se han realizado pruebas estableciendo el número de propiedades a 1, 2, 3, 4 y 5, teniendo los demás parámetros valores constantes. Estos valores constantes se han establecido con el fin de que el tiempo de razonamiento sea lo suficientemente significativo como para poder apreciarse diferencias en el rendimiento del motor de reglas.

Así, tras varias pruebas realizadas con el fin de detectar estos valores, se han establecido las siguientes configuraciones para la presente prueba, siendo el único parámetro variable el número de propiedades.

- **# propiedades: 1, 2, 3, 4 y 5**
- # instancias: 1.000
- # reglas: 100
- # condiciones: 1
- Semilla: 10

Para cada una de las cinco configuraciones variables en función del número de propiedades, se han generado las reglas, se han creado e insertado las instancias correspondientes en la Base de Conocimiento y se ha lanzado el proceso de razonamiento, obteniendo así el tiempo invertido en dicho proceso especificado en milisegundos.

Los resultados obtenidos en cada una de las ejecuciones de la presente prueba se exponen en el gráfico que se muestra a continuación.

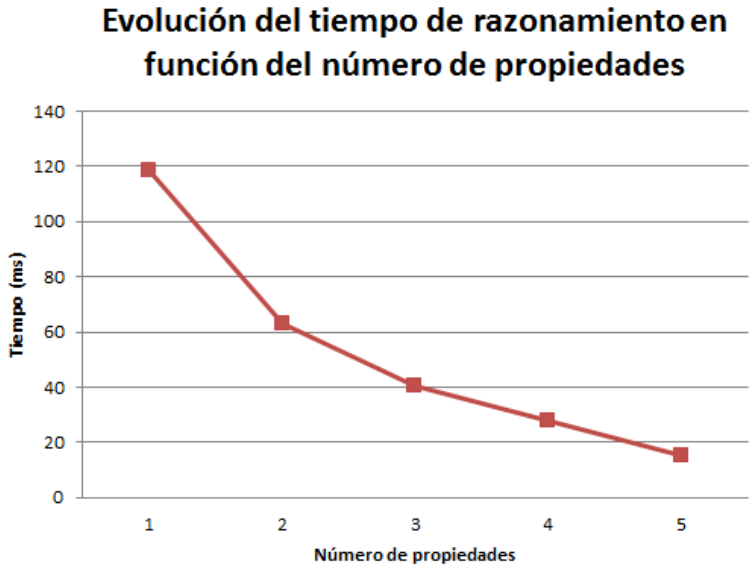


Gráfico 1. PER1 - Evolución del tiempo de razonamiento en función del número de propiedades

Como puede observarse, el tiempo empleado en el proceso de razonamiento disminuye a medida que aumenta el número de propiedades sobre las que se configuran las condiciones de la regla. Esto se debe a que cuanto mayor sea el número de restricciones impuestas en las condiciones de las reglas, más efectivo será el filtrado o discriminación que el motor de reglas realiza sobre las instancias de la Base de Conocimiento.

Además, el número de reglas cuya parte consecuente se lanza es menor cuando el número de propiedades sobre las que se establecen las condiciones de las reglas aumenta. Así, en la configuración establecida de 2 propiedades, se lanzan una media de 5.000 acciones sobre el consecuente de las reglas, mientras que con 5 propiedades utilizadas para configurar las condiciones, se lanzan una media de 112 acciones en la ejecución de las mismas reglas. Al tener que ejecutar menos acciones en el consecuente de las reglas, el tiempo que se emplea en el proceso de razonamiento también se ve reducido.

De esta manera, el mayor tiempo consumido por el motor de reglas en las pruebas realizadas es de 118 milisegundos, que se da cuando se utiliza una sola propiedad para configurar la condición de las reglas generadas. Este tiempo resulta aceptable ya que se encuentra por debajo de los 1.000 milisegundos que se toman de partida a modo de referencia.

6.1.2.2. **PER2 - Número de instancias**

La segunda prueba analiza el efecto del número de instancias almacenadas en la Base de Conocimiento sobre la ejecución del sistema. Para ello, se ha variado la configuración del número de instancias, estableciendo sus valores a 1.000, 5.000, 10.000, 15.000 y 20.000. El resto de parámetros se han configurado con valores fijos, tal y como se detalla a continuación.

- # propiedades: 2
- **# instancias: 1.000, 5.000, 10.000, 15.000 y 20.000**
- # reglas: 100
- # condiciones: 1
- Semilla: 10

Los resultados obtenidos en las pruebas realizadas se muestran en el siguiente gráfico.

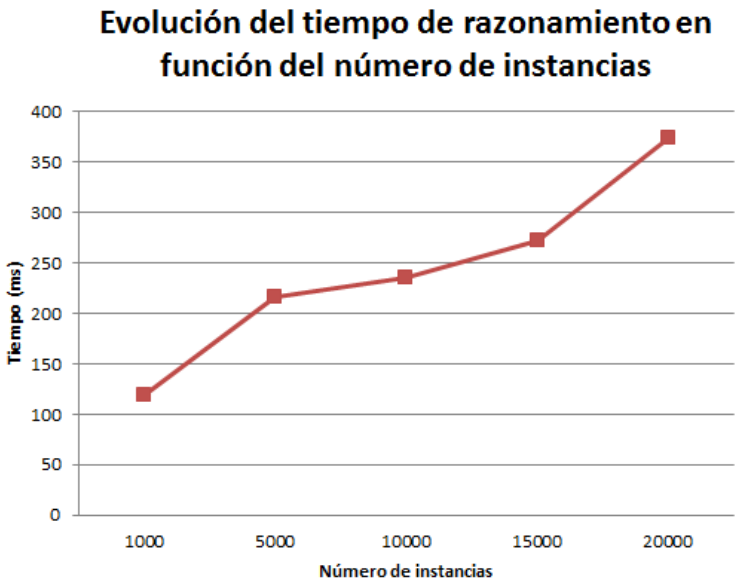


Gráfico 2. PER2 - Evolución del tiempo de razonamiento en función del número de instancias

En esta ocasión, a medida que aumenta el número de instancias almacenadas en memoria en la Base de Conocimiento, el tiempo empleado en el razonamiento incrementa, ya que el motor de razonamiento tiene que evaluar un mayor número de objetos en memoria. Además, cuantas más instancias se encuentren en la Base de Conocimiento, más acciones se ejecutarán por el cumplimiento de las reglas y más

tiempo empleará el proceso de razonamiento. Así, la media de acciones que se ejecutan cuando el número de instancias es 1.000, son 6.000 acciones, mientras que cuando las instancias son 20.000, el número de acciones aumenta hasta las 100.000, afectando así al rendimiento del razonamiento.

El mayor tiempo empleado ha sido con una configuración de 20.000 instancias, con un tiempo medio de 374 milisegundos, el cual sigue estando dentro de los límites establecidos

6.1.2.3. **PER3 - Número de reglas**

Una tercera prueba mide el rendimiento del sistema en función del número reglas. Así, se han configurado pruebas utilizando 100, 200, 300, 400 y 500 reglas. El resto de parámetros se han mantenido con valores fijos, tal y como se establece a continuación.

- # propiedades: 2
- # instancias: 1.000
- **# reglas: 100, 200, 300, 400 y 500**
- # condiciones: 1
- Semilla: 10

Los resultados obtenidos se muestran en el Gráfico 3.

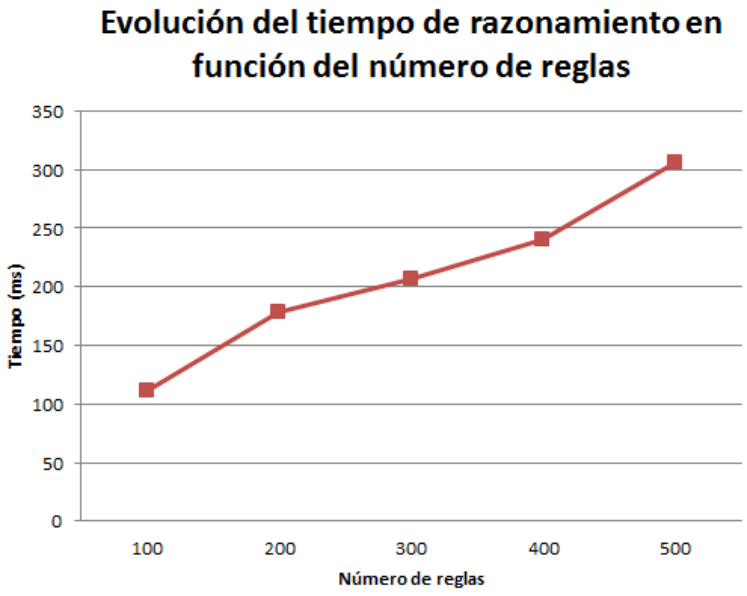


Gráfico 3. PER3 - Evolución del tiempo de razonamiento en función del número de reglas

El gráfico anterior muestra el incremento en el tiempo de razonamiento a medida que el número de reglas aumenta, ya que cuantas más reglas se establecen, más comprobaciones se tienen que realizar sobre las instancias almacenadas en la Base de Conocimiento. El mayor tiempo empleado en el razonamiento se ha obtenido con una configuración de 500 reglas, invirtiendo 306 milisegundos.

6.1.2.4. **PER4 - Número de instancias con dos condiciones**

Finalmente, se ha realizado una cuarta prueba en la que se realizan diferentes configuraciones especificando un número de instancias variable sobre la Base de Conocimiento y estableciendo a 2 el número de condiciones utilizadas. Así, las condiciones se establecen sobre entidades de tipo "Bean_A" y "Bean_B". Los parámetros configurados son los siguientes.

- # propiedades: 2
- **# instancias: 500, 1.000, 2.000, 3.000 y 4.000**
- # reglas: 100
- **# condiciones: 2**
- Semilla: 10

Los resultados obtenidos se muestran en el gráfico que se muestra a continuación.

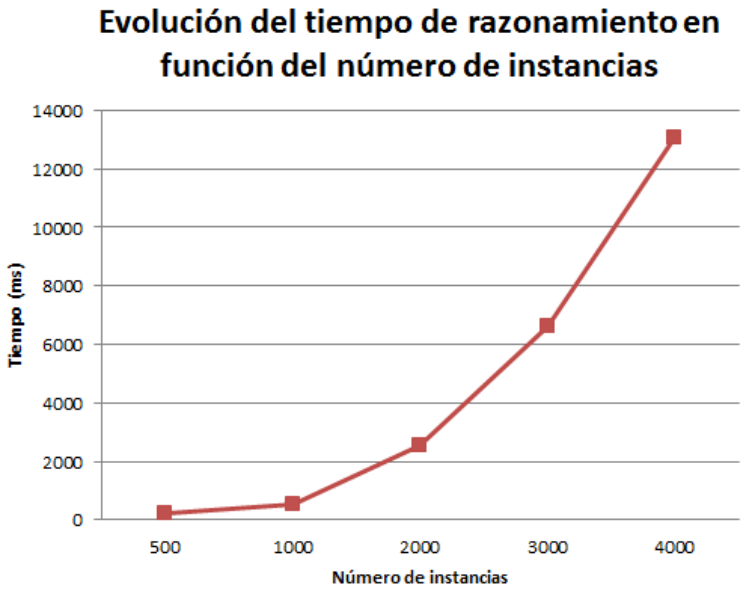


Gráfico 4. PER4 - Evolución del tiempo de razonamiento en función del número de instancias

En esta ocasión puede apreciarse cómo también el tiempo de razonamiento aumenta a medida que aumenta el número de instancias almacenadas en la base de conocimiento. Cabe destacar, que en esta ocasión los tiempos obtenidos se disparan a partir de las 1.000 instancias, donde se consigue un tiempo de razonamiento de 533 milisegundos. A partir de entonces, el tiempo de razonamiento aumenta considerablemente, llegando a los 13.049 milisegundos de la configuración con 4.000 instancias. Esto se debe a la utilización de dos tipos de entidades diferentes sobre las que se configuran las condiciones de las reglas generadas. Por lo tanto cuantas más entidades se utilicen para establecer las condiciones de la regla, mayor es el tiempo empleado en el proceso de razonamiento.

Este crecimiento es exponencial, tal y como se aprecia en la evolución del tiempo de razonamiento empleado. Además, cabe destacar que al intentar configurar la prueba para su ejecución con 5.000 instancias, la pila de memoria de la máquina virtual Java reservada para el almacenamiento de objetos (un gigabyte) se llena completamente al insertar las instancias de entidades en la Base de Conocimiento. Esto se debe a que el espacio que reserva el motor de reglas en memoria para poder lanzar los mecanismos de razonamiento pertinentes es muy elevado y no deja que el programa siga su ejecución.

6.1.2.5. **PER5 - Número de instancias con dos condiciones y cuatro propiedades**

Tomando como punto de partida la configuración anterior en la que el consumo de tiempo en el proceso de razonamiento es elevado, llegando incluso a saturar la memoria disponible, se han realizado pruebas adicionales con el fin de ver si el rendimiento obtenido puede ser mejorado realizando cambios en algunos de los parámetros especificados en la configuración de la prueba.

Así, se ha aumentado el número de propiedades a un total de 4 con el fin de ver el comportamiento del sistema de razonamiento. Las configuraciones son las siguientes.

- **# propiedades: 4**
- **# instancias: 500, 1.000, 2.000, 3.000 y 4.000**
- # reglas: 100
- **# condiciones: 2**
- Semilla: 10

Los resultados obtenidos se muestran en el siguiente gráfico.

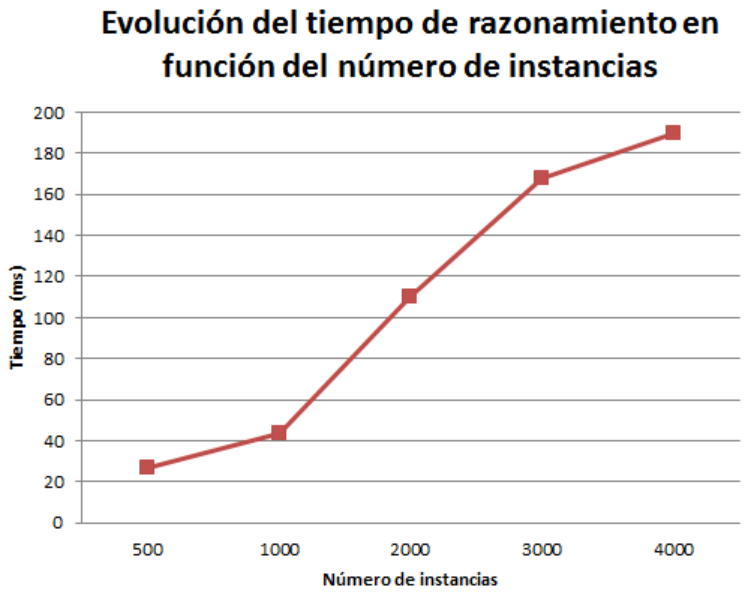


Gráfico 5. PER5 - Evolución del tiempo de razonamiento en función del número de instancias

Como puede apreciarse, el tiempo empleado en el razonamiento disminuye considerablemente aumentando el número de propiedades de las instancias de entidades utilizadas sobre las que se configuran las condiciones de las reglas, hecho que ya se constató en la prueba PER1.

Además, disminuyendo el número de reglas a evaluar, se disminuye también el tiempo de razonamiento, tal y como sucede en la prueba PER3. Por ejemplo, tomando como punto de partida la configuración de la prueba PER4 en la que el sistema de razonamiento se encuentra al borde de saturar la memoria disponible para continuar su ejecución, se ha realizado una prueba adicional. Esta vez, se ha establecido el número de reglas a 10 y se ha configurado la prueba sobre un total de 4.000 instancias para ver el comportamiento del sistema de razonamiento.

- # propiedades: 2
- **# instancias: 4.000**
- **# reglas: 10**
- **# condiciones: 2**
- Semilla: 10

Con esta configuración, el sistema emplea 3.300 milisegundos en el proceso de razonamiento, cifra que aunque no resulta muy eficiente, está lejos de los 14.000 milisegundos que tardaba el proceso de razonamiento en la prueba PER4 con una configuración de 100 reglas. Además en esta ocasión el proceso está lejos de llegar a saturar la memoria de la máquina virtual Java reservada para almacenar objetos. De esta manera, se constata el hecho de que a menor número de reglas menor es el tiempo empleado en el razonamiento, tal y como ocurría en la prueba PER3.

6.1.3. Consideraciones finales

Como se ha podido comprobar en las pruebas realizadas, el rendimiento obtenido entra dentro de lo que se ha marcado como razonable, es decir, que el tiempo de ejecución del ciclo de razonamiento sea inferior a los 1.000 milisegundos. Estos tiempos de razonamiento se han conseguido con unas configuraciones de instancias, propiedades, reglas y condiciones determinadas que se han tomado como base haciendo una simulación de lo que los usuarios pueden llegar a requerir en sus configuraciones con la plataforma. Aun así, estos escenarios de configuración pueden ser variables y no encontrarse dentro de los aquí analizados. De todas maneras, las pruebas realizadas sirven para sacar diversas conclusiones sobre la optimización del rendimiento del motor de razonamiento y algunas consideraciones a tener en cuenta por los usuarios de la plataforma en caso de que el rendimiento no sea el esperado.

También se debe tener en cuenta que el razonamiento propuesto en las pruebas diseñadas, parte de una Base de Conocimiento sobre la que previamente no se ha aplicado ningún ciclo de razonamiento. Por lo tanto, hay que tener en cuenta que en sucesivos ciclos de razonamiento el tiempo empleado puede disminuir, ya que no todas las instancias de la Base de Conocimiento pueden llegar a ser actualizadas. Así, el motor de razonamiento solamente evaluará aquellas instancias modificadas sobre las reglas ya procesadas, reduciendo el tiempo empleado en el ciclo inicial de razonamiento.

Como conclusiones generales de las pruebas realizadas, se han elaborado una serie de recomendaciones y guías generales para mejorar el rendimiento del proceso de razonamiento. Estas recomendaciones se detallan a continuación.

- En cuanto a las funciones que la plataforma ofrece para ser utilizadas en el consecuente de las reglas, es importante que no consuman mucho tiempo de ejecución y que la invocación de dichas funciones se realice de manera

asíncrona para que afecte en menor medida al proceso de razonamiento. Así, todas las funciones disponibles en la plataforma han sido implementadas de manera que su procesamiento se realice en un hilo de ejecución diferente al del propio proceso de razonamiento.

- Cuantas más propiedades de las entidades definidas se utilicen para configurar las condiciones de las reglas, mejor será el rendimiento del proceso de razonamiento (PER1). De esta manera, resulta importante concretar con el máximo detalle, las condiciones de las que consta cada situación configurada mediante las reglas, ya que cuantas más condiciones se dispongan sobre los valores de contexto utilizados, menor será el tiempo empleado en el ciclo de razonamiento.

- Cuantas más instancias se dispongan en la Base de Conocimiento, mayor será el tiempo empleado en el ciclo de razonamiento (PER2). Por lo tanto, hay que prestar atención al número de entidades precisadas para modelar la información de contexto y la cantidad de instancias que se van a generar de dichas entidades para ser almacenadas.

- Cuantas menos reglas se configuren, mejor será el rendimiento del ciclo de razonamiento (PER3). De esta manera, hay que intentar que el número de reglas sea lo más razonable posible.

- Cuantas más entidades se combinen en las condiciones de una regla, peor será el tiempo de razonamiento (PER4). Este caso es importante, ya que cuantas más entidades se inserten en la Base de Conocimiento, el tiempo empleado en cada ciclo de razonamiento aumentará exponencialmente, llegando incluso a saturar la memoria consumida por el motor de razonamiento. En caso de combinar diferentes entidades en las reglas, es importante utilizar el mayor número de restricciones posibles sobre las propiedades de las mismas, con el fin de que el ciclo de razonamiento tenga un tiempo de ejecución inferior (PER5).

- El consumo de memoria utilizado por el motor de reglas es elevado, ya que todo el proceso se realiza en memoria. Por lo tanto, es importante reservar el máximo posible de memoria destinada al almacenamiento de objetos en la máquina virtual Java del servidor donde esté desplegada la plataforma.

6.2. Evaluación de usuario

Esta sección recoge la evaluación que se ha realizado tanto de la metodología para el desarrollo colaborativo de sistemas sensibles al contexto como de la plataforma para realizar dichos desarrollos en función de la metodología. Esta evaluación se ha realizado bajo la premisa colaborativa entre expertos en el dominio y programadores. En siguientes secciones se detallan la metodología que se ha seguido para llevar a cabo la evaluación, las pruebas realizadas y los resultados obtenidos.

6.2.1. Metodología

Tanto la metodología para el desarrollo de sistemas sensibles al contexto como la propia plataforma de desarrollo han sido validadas con un grupo de 20 personas en un entorno controlado. La mitad de este grupo ha estado conformado por personas con conocimientos de programación y la otra mitad ha estado compuesta por expertos en el dominio turístico[49] que poseían pocos o nulos conocimientos de programación.

De esta manera, las pruebas se han llevado a cabo en parejas formadas por un experto en el dominio y un técnico programador, donde el objetivo ha consistido en realizar un trabajo colaborativo para resolver una serie de tareas siguiendo los pasos establecidos en la metodología de desarrollo y por mediación de la plataforma.

El escenario donde se ha enmarcado la prueba de evaluación con el grupo de usuarios ha sido un entorno laboral relacionado con el dominio turístico. Así, se ha simulado un escenario donde el experto en el dominio turístico ha hecho las veces de técnico de una oficina de turismo, con la necesidad de desplegar un servicio móvil entre los visitantes, que sea capaz de proporcionar información adaptada al contexto de los mismos. El programador ha sido, por su parte, el agente que ha dispuesto de la plataforma que posibilita el modelado de las situaciones requeridas por el técnico de turismo.

Tanto la metodología como la propia plataforma han sido utilizadas para poder involucrar durante la prueba de evaluación al experto en el dominio en el proceso de desarrollo junto al programador.

[49] Se han considerado como expertos en el dominio turístico aquellas personas cuya ocupación laboral está o ha estado relacionada con el sector turístico.

En siguientes secciones se presentan los datos de los participantes que realizaron la prueba de evaluación, la descripción de la prueba realizada y los fundamentos que se han tenido en cuenta para diseñar el cuestionario que los participantes han tenido que completar una vez finalizada la prueba de evaluación.

6.2.1.1. **Participantes**

El 40% de los participantes fueron mujeres. Los rangos de edad se han dividido en diferentes franjas, siendo la franja predominante la correspondiente a las edades entre los 26 y 30 años, tal y como se muestra en el siguiente gráfico.

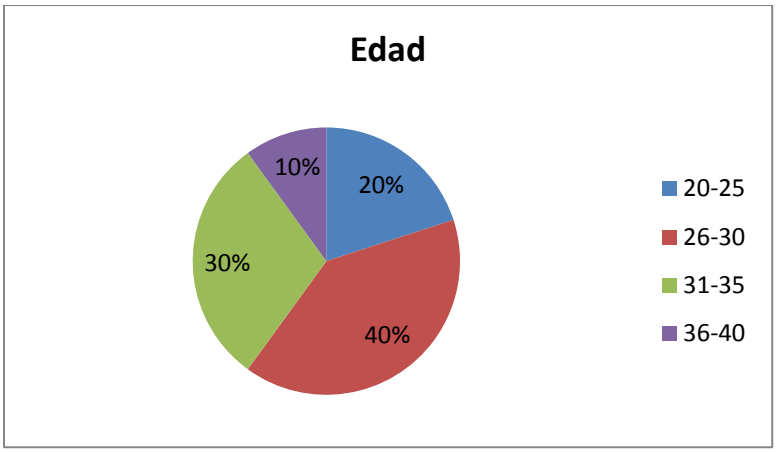


Gráfico 6. Rangos de edad de los participantes

Atendiendo a la formación académica, en su mayoría poseían títulos universitarios, con un 90% del total de participantes.

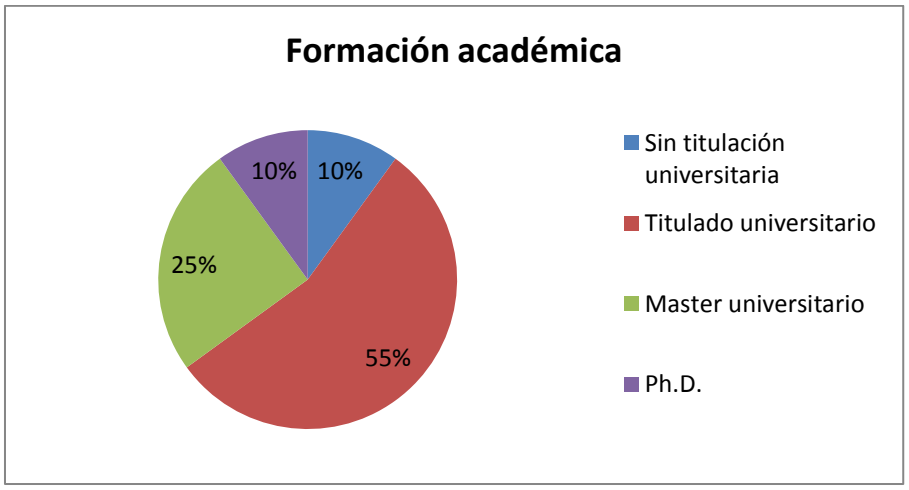


Gráfico 7. Formación académica de los participantes

En cuanto a la profesión de los participantes, un 75% de los mismos aseguró trabajar en empresas enmarcadas en el sector de las Tecnologías de la Información y

Comunicación (TIC), por lo que ciertos expertos en el dominio también tenían algún tipo de experiencia en el sector tecnológico.

Gráfico 8. Profesión de los participantes

En cuanto al grado de conocimiento en temas relacionados con el mundo de las tecnologías y más concretamente, con la programación de software, el 50% de los participantes de la prueba de evaluación afirmaban tener un nivel alto, tal y como se refleja en el Gráfico 9. Este porcentaje se debe a que la mitad de los participantes de la prueba han desempeñado en algún momento labores de programación en sus respectivos trabajos.

Además, dentro del 50% restante de participantes, donde se enmarcan los expertos en el dominio, existen perfiles con algún tipo de conocimiento de programación. Esto se debe a la relación laboral de algunos de los participantes con el mundo de las tecnologías de la información y comunicación.

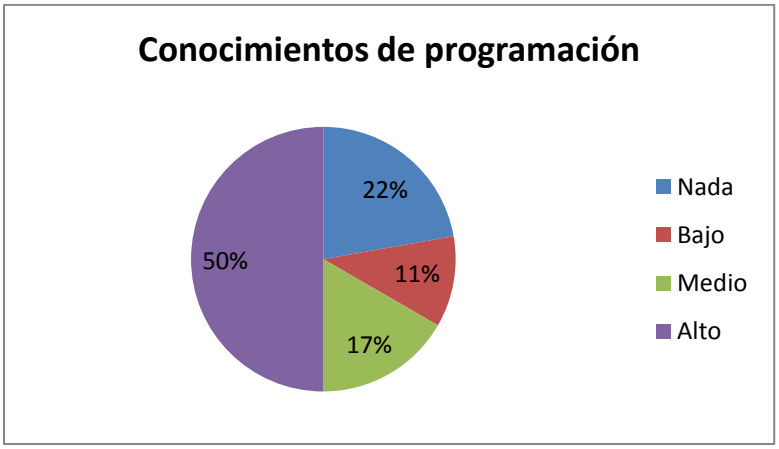


Gráfico 9. Conocimientos de programación de los participantes

6.2.1.2. **Descripción de la prueba**

En primer lugar, todos los participantes fueron informados de la prueba que iban a realizar. Además, se les impartió un curso de una hora de duración[50], en el que se exponía la base teórica sobre la que se sustenta la plataforma de desarrollo, y se les explicaba el funcionamiento de la metodología, además de mostrar las funcionalidades de la plataforma *Context Cloud*. Posteriormente, y en parejas, tuvieron que hacer frente a la prueba, la cual tenía una duración máxima de dos horas.

El enunciado de la prueba entregado a los participantes relata un escenario turístico en el que un visitante se encuentra de ruta por la ciudad realizando una serie de actividades. Además, el visitante dispone de una guía móvil turística que es capaz de proporcionar información en función de la situación en la que se encuentra.

A continuación se muestra el texto del enunciado de la prueba en el que describe el comportamiento del visitante en movilidad sobre el que los participantes tuvieron que identificar las cuatro situaciones marcadas en negrita para ser analizadas y configuradas en colaboración utilizando la metodología y la plataforma.

Son las 14 horas y David ha terminado de comer. Decide ir a la Playa de Gros a tomar el sol y darse un buen baño, por lo que sale del Hotel para dirigirse a la Parada de Bus 1 donde cogerá un autobús que le lleve al centro de la ciudad para después continuar andando hasta la playa. ***Una vez en la parada, recibe un aviso en su dispositivo móvil informándole de que el autobús número 1, con dirección al centro de la ciudad, pasa a "y media" y a "en punto" por esa parada.***

Una vez que llega al centro, se baja en la Parada de Bus 2 y camina hasta la Playa de Gros. Una vez allí, se tumba a tomar el sol. ***Su dispositivo móvil le advierte de que hace una temperatura superior a los 30 grados, por lo que debe echarse protección solar por todo el cuerpo si no quiere quemarse.***

Son las 17:00 y David decide ir de compras antes de volver al hotel. Llegadas las 19:00 horas y tras una agotadora jornada de tiendas, decide volver al hotel para poder cenar, por lo que se dirige a la Parada de Bus 2 a esperar al autobús que le lleve de vuelta. ***Cuando está esperando, el móvil le informa de que el autobús pasa por esa parada "a menos cuarto" y a "y cuarto"****.*

[50] Se puede encontrar la versión online del curso impartido en el siguiente enlace https://vimeo.com/contextcloud

Una vez en el hotel, se dirige al comedor, que está abierto entre las 20:00 y 22:00 horas para el turno de cenas. Cuando termina de cenar, se dirige a su cuarto para descansar. ***Al entrar en su habitación, advierte que el sistema de aire acondicionado se enciende de manera automática, activándose además el "modo nocturno" manteniendo la habitación a una temperatura ideal para dormir****. Esto se debe a que el sistema de gestión inteligente de habitaciones del hotel, detectó a David al entrar en el mismo y accionó el sistema con antelación.*

Una situación se daba por resuelta o detectada cuando la plataforma mostraba en la consola de depuración (Sección 5.2.2) un mensaje advirtiendo de que la situación había sido detectada por la plataforma.

Para la realización de la prueba se facilitó a los participantes un manual de la metodología de desarrollo basado en los contenidos del Capítulo 4 y un manual de la plataforma basado en los contenidos del Capítulo 5. Además, se les entregó una hoja Excel en la que figuraba la plantilla para dar soporte a la fase de análisis (Sección 4.1.1.3), y poder así identificar y parametrizar las diferentes situaciones planteadas en la prueba.

El escenario descrito fue simulado mediante la herramienta *Siafu*[51], que permite la simulación de entornos con personas en movilidad además de la generación de información de contexto derivada de dicha movilidad. De esta manera, se implementó el escenario descrito sobre el motor de simulación proporcionado por la herramienta, estableciendo todos los puntos por los que se desplaza el visitante sobre un plano de la ciudad de Donostia-San Sebastián.

En la siguiente figura se muestra la simulación implementada y los puntos sobre los que se produce la movilidad del visitante, los cuales corresponden a las localizaciones en las que se tienen que detectar las situaciones.

[51] http://siafusimulator.sourceforge.net/ Último acceso 15-05-2012.

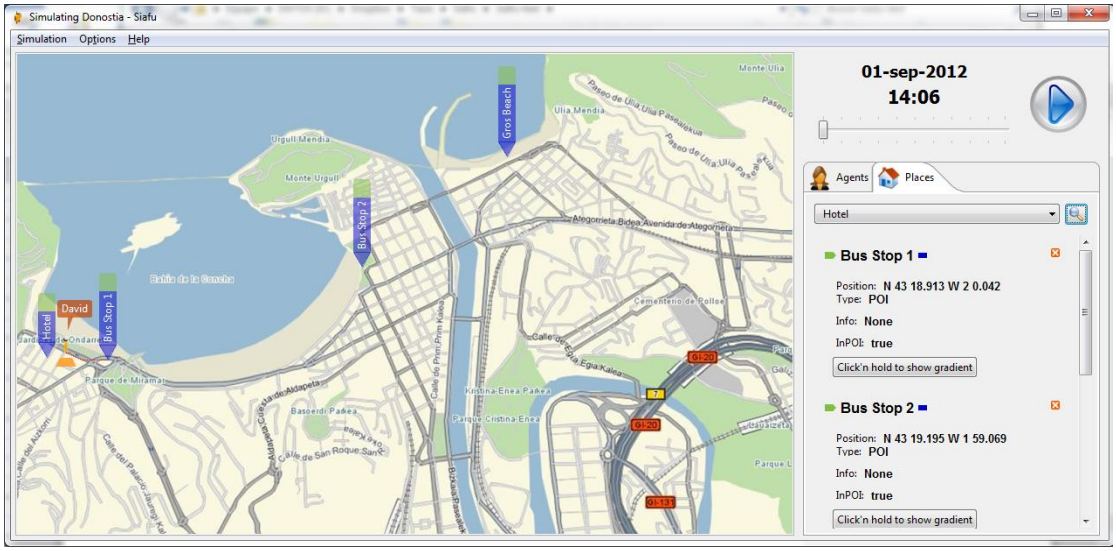


Fig. 84. Simulador de contexto

El simulador dispone de un control para iniciar la simulación y un control con el que regular la velocidad de la misma. El simulador puede configurarse para enviar la información de contexto generada a la plataforma a través de proveedores configurados en la misma. Concretamente, los datos de contexto que el simulador es capaz de generar y enviar a la plataforma una vez configurada, son los siguientes.

- Localización del visitante (latitud y longitud).
- Hora del dispositivo móvil.
- Número de teléfono del dispositivo del visitante.
- Velocidad a la que se desplaza el visitante.

El simulador proporciona además un servicio web para poder consultar las condiciones de temperatura del escenario simulado (dato necesario para identificar la segunda situación), y un servicio web que simula un detector de presencia en la habitación del hotel del visitante, que devuelve el número de personas que se encuentran en la habitación (dato necesario para detectar la situación número cuatro). Los participantes también recibieron un manual de usuario del simulador, el cual se encuentra en el Anexo III.

Durante el transcurso de las pruebas realizadas por cada una de las parejas, un observador que se encontraba en la estancia fue anotando todos los diferentes aspectos relativos a la realización de la propia prueba, además de los problemas y comentarios que los participantes realizaban mientras estaban trabajando en la resolución de las

tareas para su posterior análisis. También se anotaron tanto el tiempo total invertido en la resolución de la prueba, como los tiempos empleados en la configuración de la plataforma para cada una de las cuatro situaciones de las que consta la prueba.

El equipo utilizado para el desarrollo de la prueba ha sido un ordenador *HP TouchSmart* de 23 pulgadas. El lugar donde se realizó la prueba se presenta en la siguiente figura, donde se muestra el equipo utilizado dispuesto sobre una mesa alta donde los participantes pudieron trabajar de manera colaborativa para la resolución de la misma.

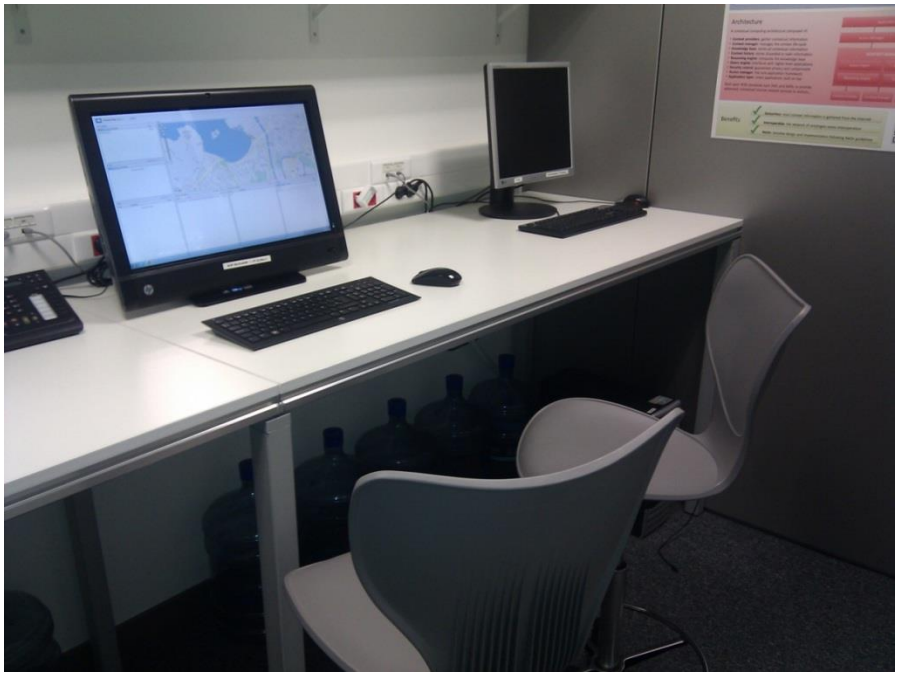


Fig. 85. Lugar de realización de las pruebas

6.2.1.3. **Cuestionario de evaluación**

Al finalizar la prueba, los participantes tuvieron que rellenar un cuestionario de evaluación. Para ello se ha diseñado una evaluación basada en modelos de aceptación tecnológica, los cuales se centran en comprender la predisposición de las personas a utilizar una determinada tecnología en un entorno laboral. Uno de los modelos de mayor influencia en este campo es el denominado *Technology Acceptance Model (TAM)*, el cual es un modelo motivacional de usuario que examina los efectos de las características de un sistema (funcionalidad y características de la interfaz) a través de la *Facilidad de uso* y la *Utilidad percibida* en la *Intención de los usuarios de utilizar un sistema* (Davis, 1989). Así, los principales constructos del modelo se definen de la siguiente manera.

- *Facilidad de Uso Percibida (Perceived Ease of Use - PEOU)*: grado de esfuerzo que una persona percibe al estar usando un sistema particular.
- *Utilidad Percibida (Perceived Usefulness - PU)*: grado en el que una persona cree que usando un sistema particular reforzaría su actuación en el trabajo.
- *Intención de conducta (Behavioural Intention - BI)*: intención de comportamiento ante un sistema concreto.

En base a estos constructos, se ha realizado la evaluación de la aceptación tecnológica, con el fin de comprobar que la plataforma y la metodología de desarrollo asociada a la misma son válidas para cumplir los objetivos marcados.

El cuestionario diseñado tiene un total de 20 afirmaciones, las cuales tuvieron que ser valoradas por los participantes utilizando una escala de 1 (en total desacuerdo) a 6 (totalmente de acuerdo).

En cuanto al constructo relativo a la facilidad de uso percibida, las afirmaciones que debían ser valoradas por los participantes eran las siguientes.

1. Aprender a usar el sistema ha sido fácil para mí.
2. Aprender la metodología de desarrollo ha sido fácil para mí.
3. He podido hacer con el sistema lo que quería en cada momento.
4. Es un sistema fácil de manejar y de interactuar con él.
5. Sería fácil llegar a ser un usuario experto en el manejo del sistema.
6. El sistema facilita el trabajo colaborativo.
7. La metodología de desarrollo facilita el trabajo colaborativo.
8. En general, tanto la metodología como el sistema son sencillos de usar.

En relación a la utilidad percibida tanto de la plataforma como de la metodología, las afirmaciones a valorar eran las siguientes.

9. Utilizar el sistema me ayudaría a realizar más rápido mi trabajo en el desarrollo de sistemas sensibles al contexto.
10. Utilizar el sistema mejoraría el rendimiento en mi trabajo en el desarrollo de sistemas sensibles al contexto.

11. Utilizar el sistema mejoraría la productividad en mi trabajo en el desarrollo de sistemas sensibles al contexto.
12. Utilizar el sistema aumentaría la efectividad en mi trabajo en el desarrollo de sistemas sensibles al contexto.
13. Utilizar el sistema facilitaría mi trabajo en el desarrollo de sistemas sensibles al contexto.
14. Es un sistema útil para mi trabajo.
15. El sistema es útil para realizar un trabajo colaborativo.
16. La metodología de desarrollo es útil para trabajar con el sistema.
17. La metodología de desarrollo es útil para realizar un trabajo colaborativo.
18. La metodología colaborativa es útil para implementar sistemas sensibles al contexto.

Finalmente, en relación a la intención de conducta, se realizaban las siguientes afirmaciones para su valoración.

19. Recomendaría el sistema a otros usuarios.
20. Utilizaría el sistema en futuros desarrollos de software.

Además de las anteriores afirmaciones, se preguntaba a los participantes si pagarían por el sistema y en caso afirmativo, la cantidad que pagarían.

Es interesante resaltar que para la evaluación planteada hay que tener en cuenta que la plataforma y la metodología, están pensadas para ser utilizadas en un entorno laboral. El cuestionario utilizado puede encontrarse en el Anexo IV.

6.2.2. Resultados

A continuación, se muestran los resultados de los cuestionarios realizados por cada uno de los participantes, además de los resultados del análisis de tiempos empleados por los mismos en la resolución de la prueba, las observaciones realizadas en el transcurso de la misma y los comentarios anotados.

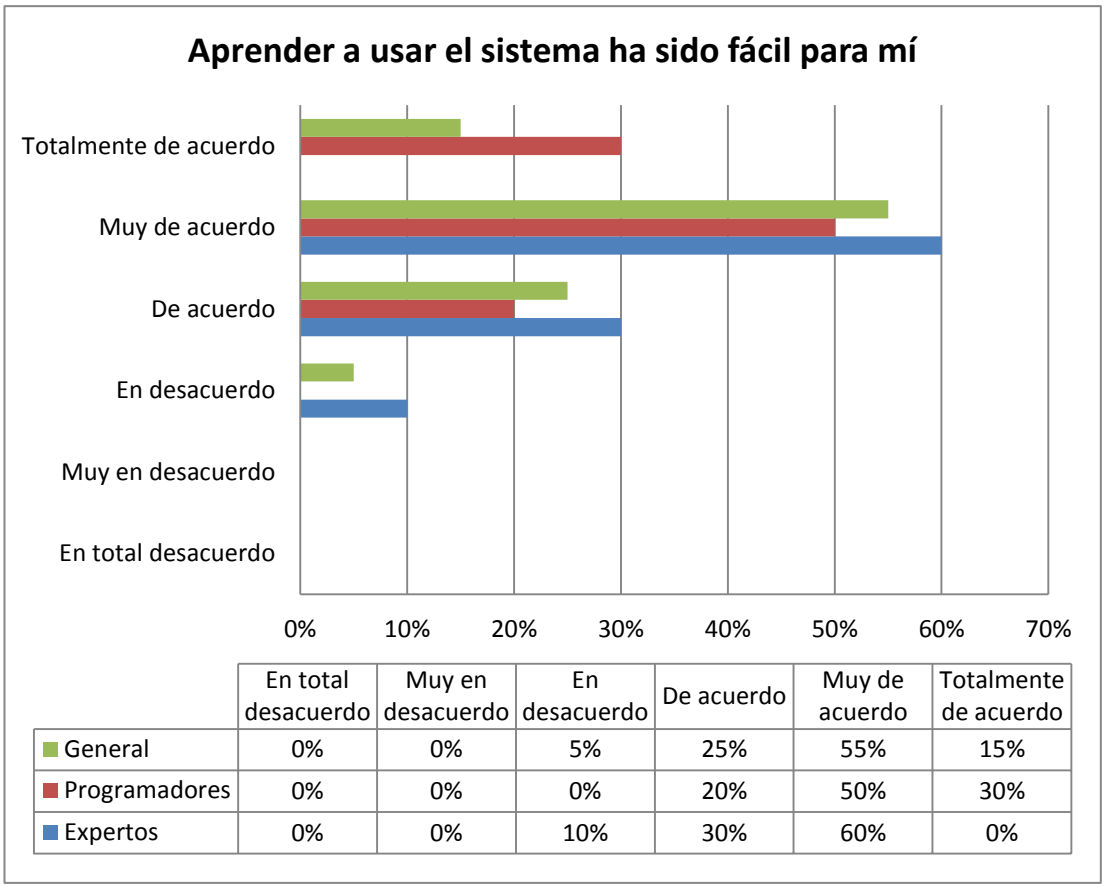
Estos resultados se dividen en seis grandes bloques referentes a los constructos utilizados y los diferentes parámetros monitorizados durante la prueba: facilidad de uso percibida, utilidad percibida, intención de conducta, observaciones, comentarios y

tiempos. En siguientes secciones se presentan los resultados obtenidos por cada uno de los bloques mencionados anteriormente.

6.2.2.1. **Facilidad de Uso Percibida**

El primer bloque de afirmaciones que componen el cuestionario repartido a los participantes comprende aquellas cuestiones relacionadas tanto con la facilidad de uso de la plataforma como de la metodología de desarrollo. En esta sección se muestran los resultados de las valoraciones recogidas sobre este constructo compuesto por un total de 8 afirmaciones.

Por cada una de las valoraciones emitidas por los participantes, se muestran los resultados de los programadores en comparación con los resultados de los expertos en el dominio, así como los resultados a nivel general.

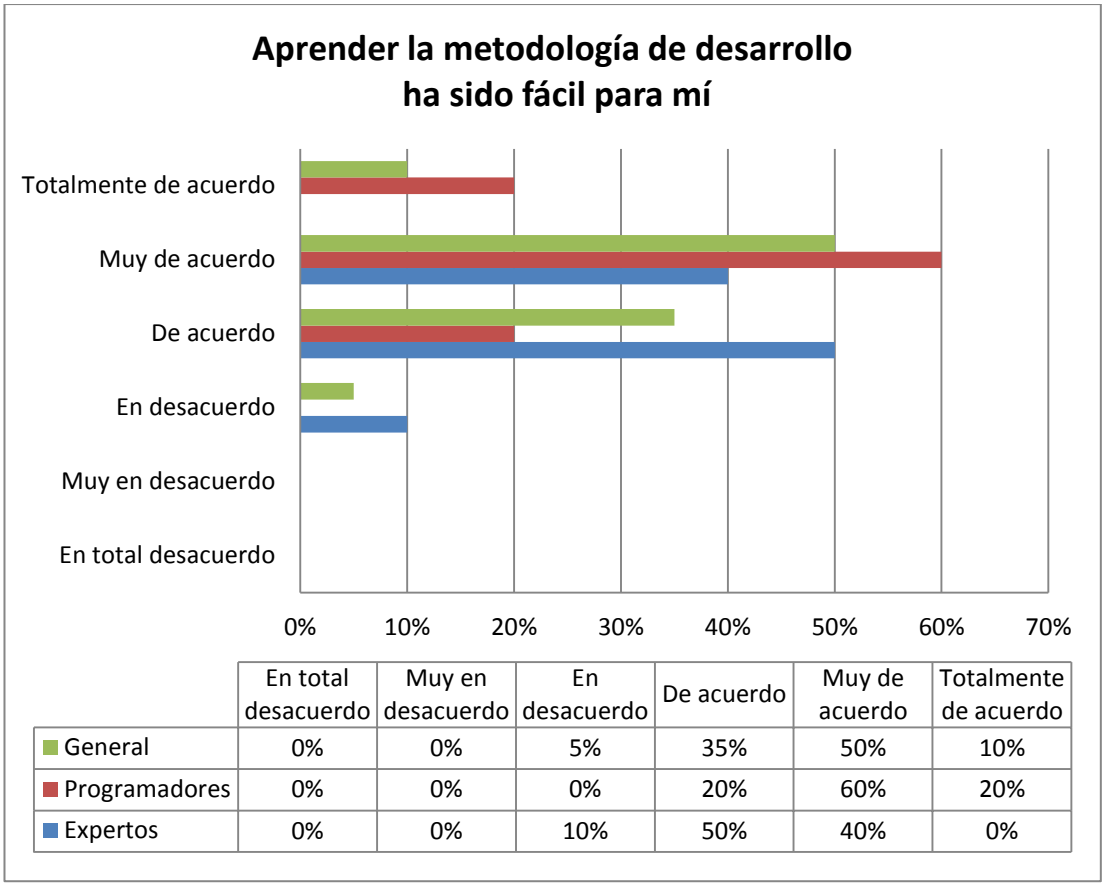


	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	5%	25%	55%	15%
Programadores	0%	0%	0%	20%	50%	30%
Expertos	0%	0%	10%	30%	60%	0%

Gráfico 10. Aprender a usar el sistema ha sido fácil para mí

En el Gráfico 10 se muestran los resultados referentes a la facilidad de aprender a utilizar el sistema, en este caso, la plataforma de desarrollo. Se puede apreciar que una amplia mayoría de los participantes están de acuerdo con dicha afirmación en cierto

grado (95%), estando la mayoría de ellos (55%) muy de acuerdo con la misma. Cabe destacar que los programadores han aprendido a utilizar la herramienta de una manera más sencilla debido a que se sienten familiarizados con este tipo de tareas y entornos. Así, existe un 30% de programadores que están totalmente de acuerdo con esta afirmación. Por contra, no existe ningún experto en el dominio que esté totalmente de acuerdo con la misma. Sin embargo, son los expertos en el dominio los únicos que se muestran en desacuerdo en un 10% de las ocasiones. Esto evidencia que existen algunos expertos en el dominio a los que les ha resultado más complicado aprender a utilizar la plataforma de desarrollo.

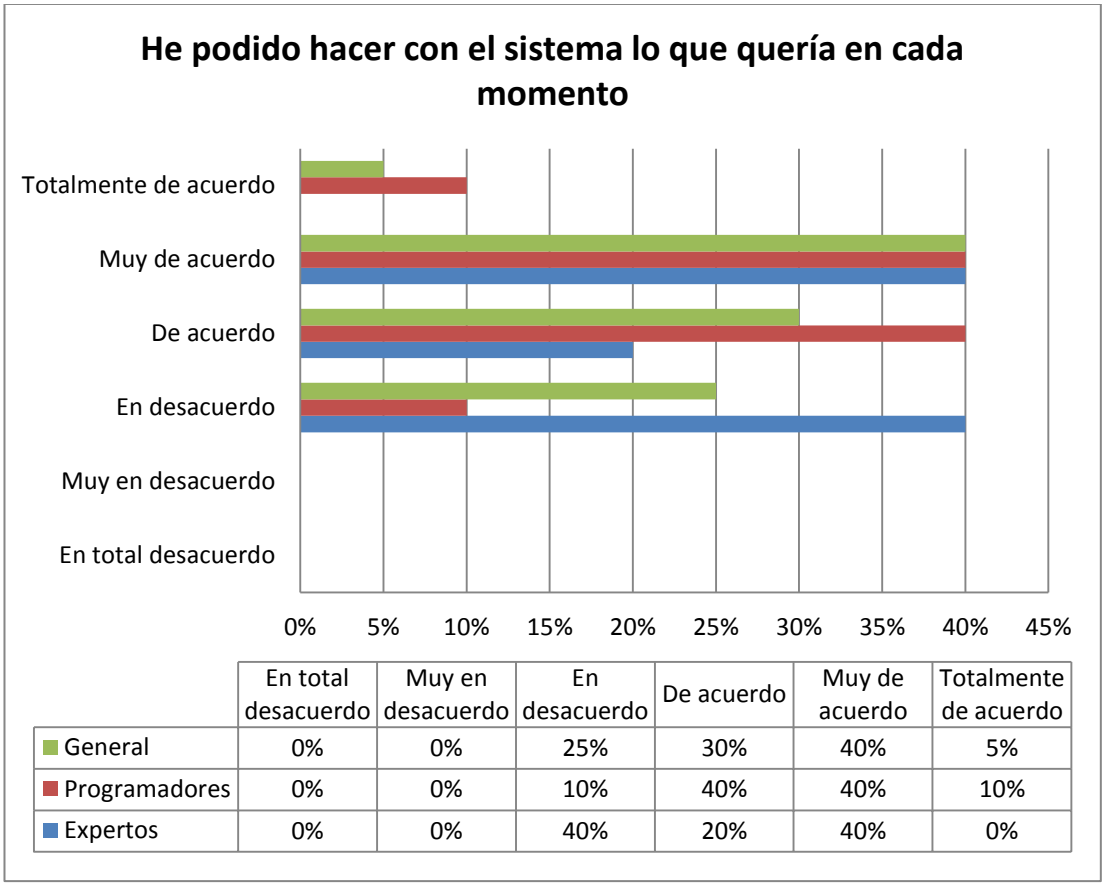


	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	5%	35%	50%	10%
Programadores	0%	0%	0%	20%	60%	20%
Expertos	0%	0%	10%	50%	40%	0%

Gráfico 11. Aprender la metodología de desarrollo ha sido fácil para mí

En cuanto al aprendizaje de la metodología de desarrollo, una amplia mayoría está en cierto grado también de acuerdo con que ha sido fácil (95%), contando con un 50% de participantes que se muestran muy de acuerdo con dicha afirmación. Aquí también cabe destacar que los programadores han aprendido con más facilidad la metodología de desarrollo que los expertos en el dominio, ya que existe un 20% de los programadores que están totalmente de acuerdo con la afirmación, mientras que no existe ningún

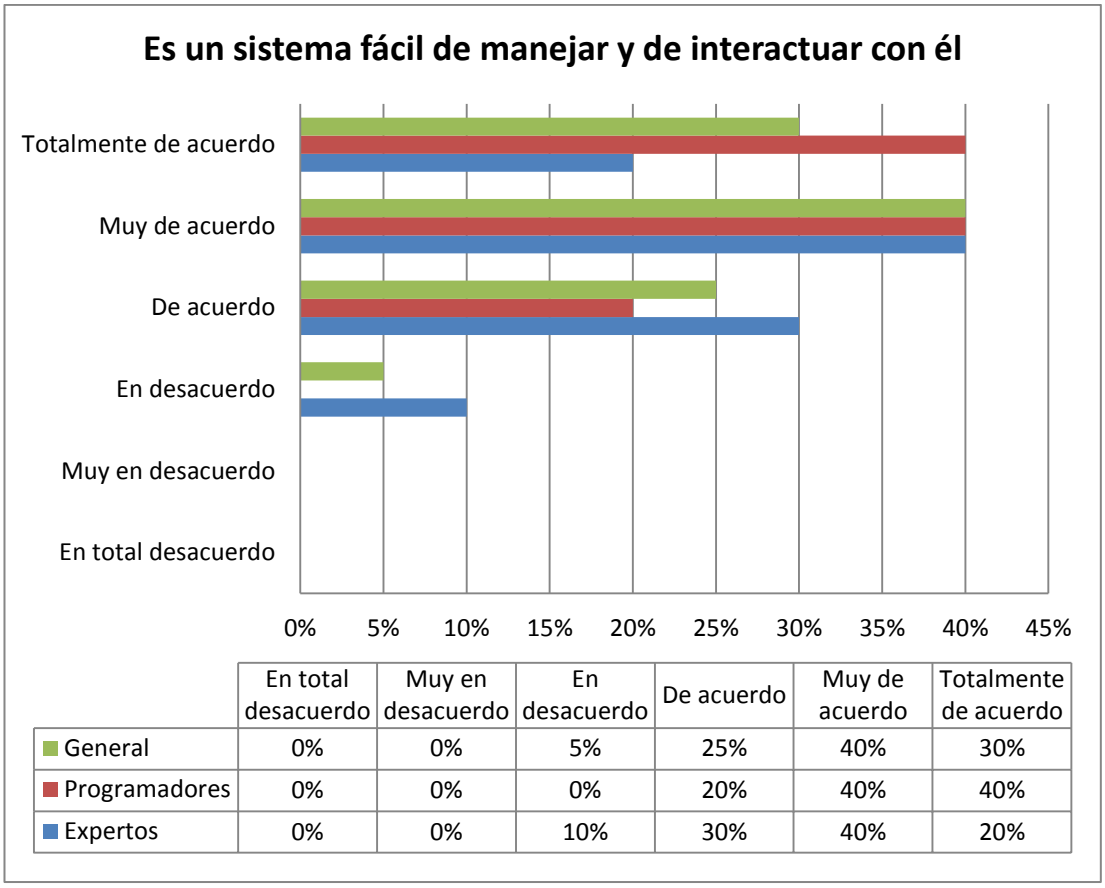
experto en el dominio que esté totalmente de acuerdo con esta opinión. Un 10% de los expertos en el dominio indica que están en desacuerdo con que la metodología haya sido fácil de aprender, por lo que existen casos en los que los expertos en el dominio no consiguen aprender con facilidad la metodología de desarrollo.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	25%	30%	40%	5%
Programadores	0%	0%	10%	40%	40%	10%
Expertos	0%	0%	40%	20%	40%	0%

Gráfico 12. He podido hacer con el sistema lo que quería en cada momento

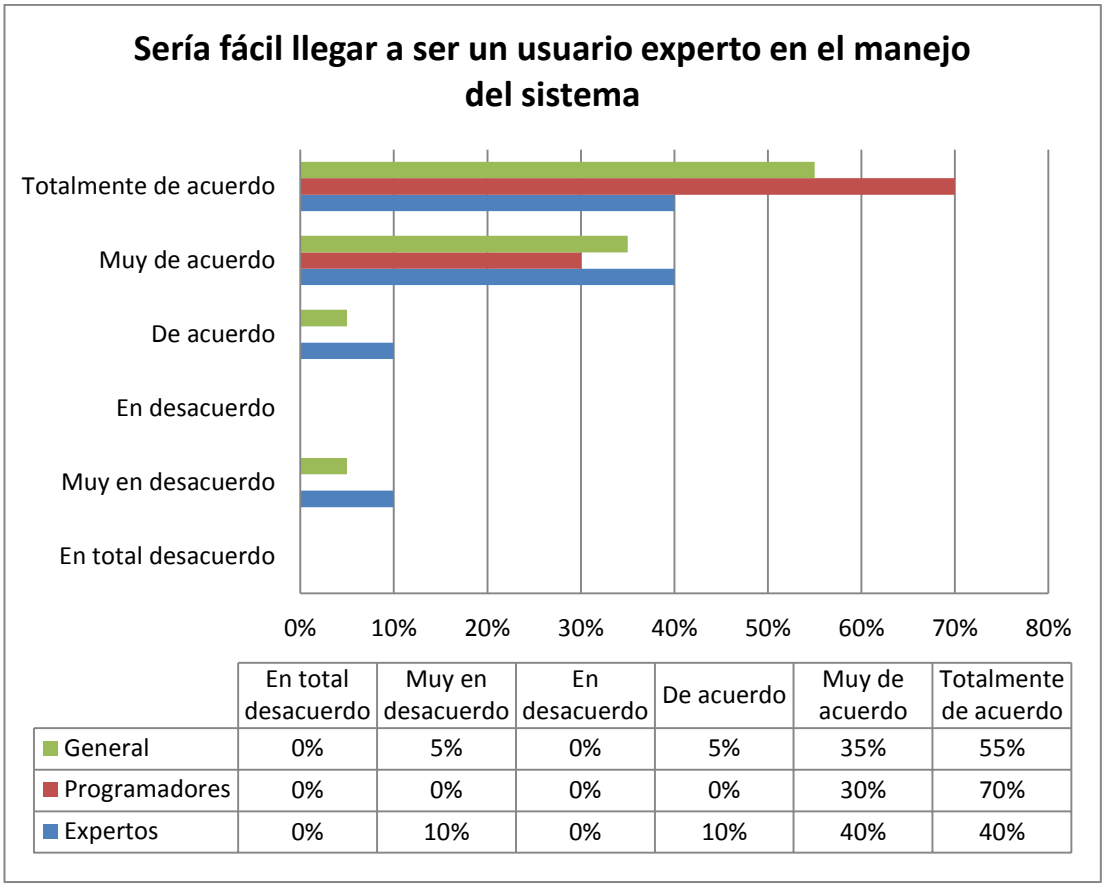
El Gráfico 12 muestra la opinión de los participantes en relación al uso del sistema. En términos generales, un 75% de los participantes afirma estar en cierto grado de acuerdo con que ha podido hacer con el sistema lo que quería en cada momento. En esta ocasión, existe un desequilibrio claro entre los programadores y los expertos en el dominio. Así, un 10% de los programadores está totalmente de acuerdo con dicha afirmación, mientras que un 40% de los expertos en el dominio está en desacuerdo con la misma. Por lo tanto, los expertos en el dominio, al no estar familiarizados con entornos de desarrollo, y menos aún, con sistemas noveles como el aquí presentado, pueden encontrarse desorientados en la interfaz de usuario en relación a las funcionalidades proporcionadas por los diferentes controles de la misma.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	5%	25%	40%	30%
Programadores	0%	0%	0%	20%	40%	40%
Expertos	0%	0%	10%	30%	40%	20%

Gráfico 13. Es un sistema fácil de manejar y de interactuar con él

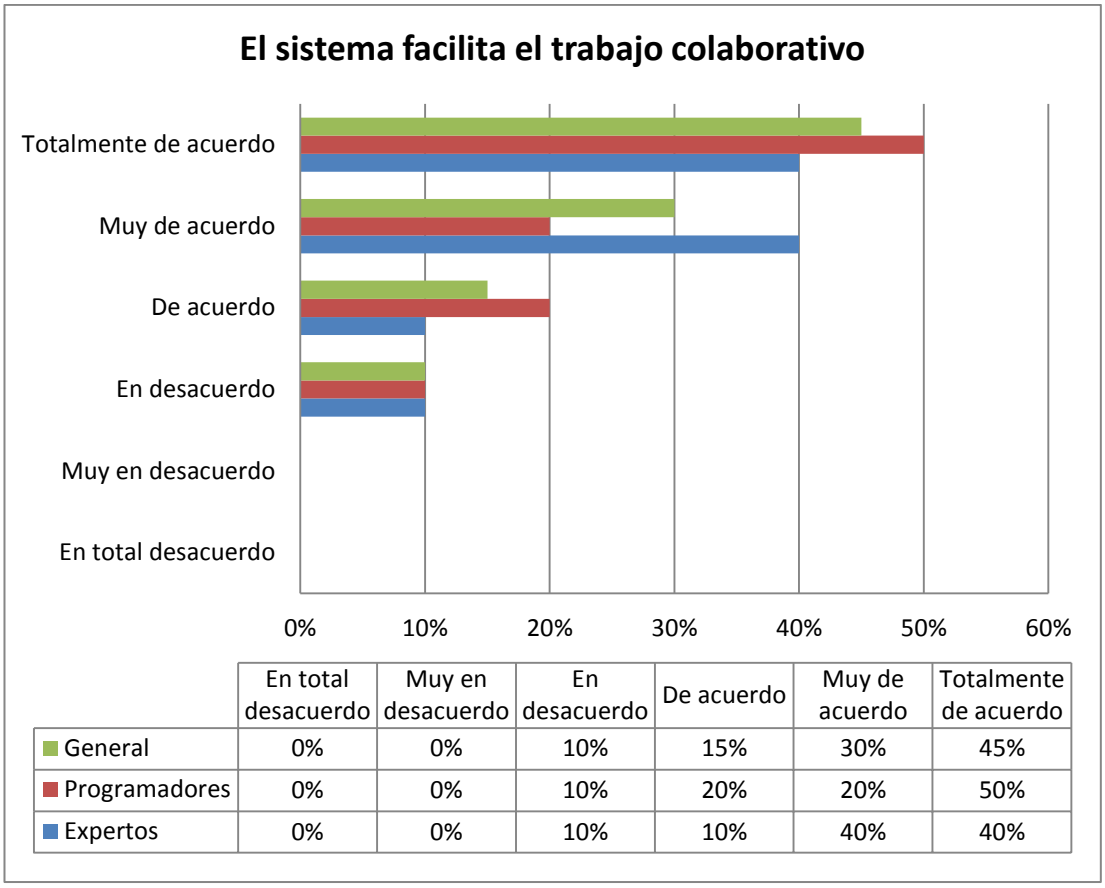
En cuanto a la facilidad de manejo de la plataforma e interacción con la misma, el Gráfico 13 muestra una amplia mayoría de participantes (95%) que están en cierta medida de acuerdo con dicha afirmación. Existe un 40% de programadores y otro 40% de expertos en el dominio que consideran que es un sistema fácil de manejar. Aun así, existe también un 10% de los expertos en el dominio que se encuentran en desacuerdo con dicha afirmación. Este porcentaje, aunque no es tan alto como el 40% de los expertos en el dominio que afirman estar en desacuerdo con que han podido hacer lo que querían en cada momento con el sistema, viene a corroborar que son los expertos en el dominio los que más dificultades han encontrado a la hora de manejar la plataforma de desarrollo, tal y como era de esperar.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	5%	0%	5%	35%	55%
Programadores	0%	0%	0%	0%	30%	70%
Expertos	0%	10%	0%	10%	40%	40%

Gráfico 14. Sería fácil llegar a ser un usuario experto en el manejo del sistema

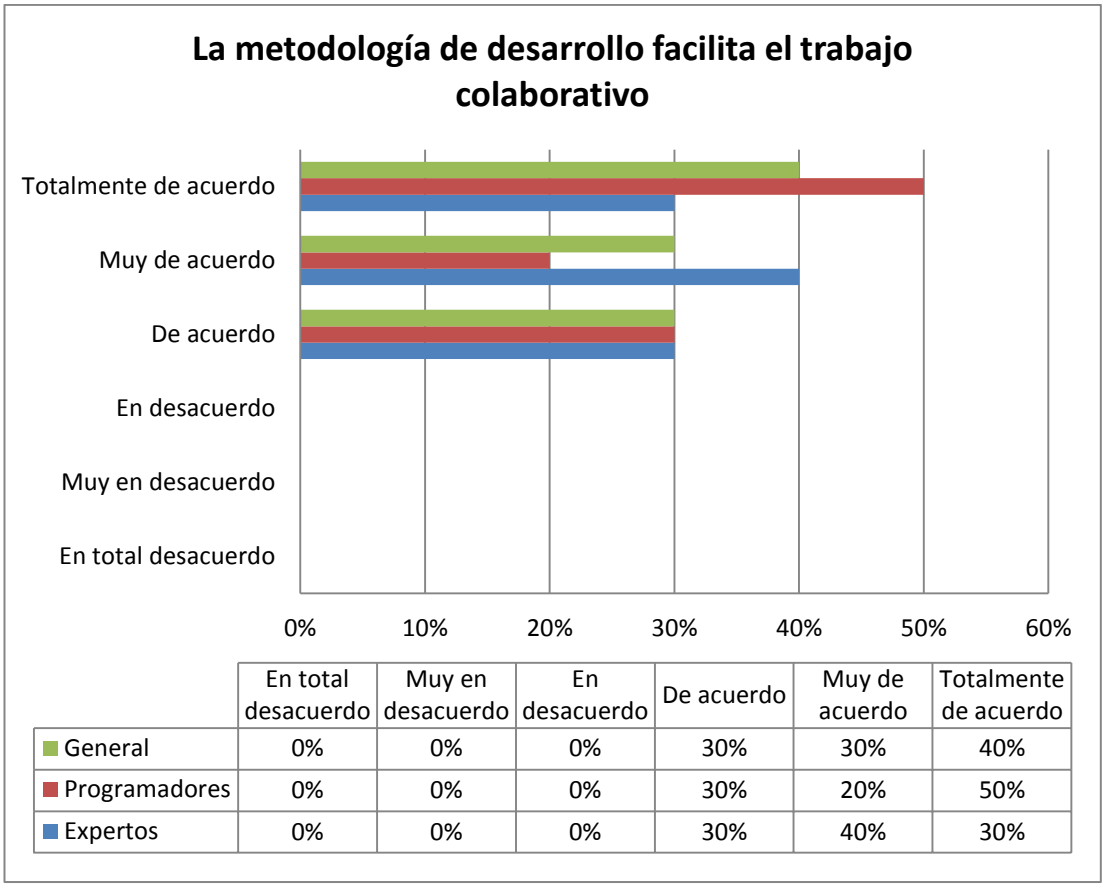
En el Gráfico 14 se muestran las respuestas en relación a la afirmación que dice que sería fácil llegar a ser un usuario experto en el manejo del sistema. En términos generales, la mayoría de participantes están en cierto grado de acuerdo con la misma (95%). Los programadores están totalmente de acuerdo en un 55% o muy de acuerdo con la misma en un 35%, lo cual implica que una amplia mayoría de los mismos estima que podría llegar a ser experto en el manejo del sistema de una manera sencilla y sin demasiado esfuerzo. Por su parte, existe también un alto porcentaje de expertos en el dominio que afirman que están totalmente de acuerdo (40%) o muy de acuerdo (35%) con que podrían ser expertos en el manejo del sistema de una manera sencilla. Estos resultados evidencian que a medida que los usuarios de la plataforma la utilizan con mayor frecuencia, ganan en seguridad en el manejo de la misma y resulta por lo tanto sencillo utilizar de manera correcta las funcionalidades ofrecidas por la plataforma. Sin embargo, sigue habiendo un mínimo porcentaje perteneciente a los expertos en el dominio que asegura estar muy en desacuerdo con esta afirmación (10%).



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	10%	15%	30%	45%
Programadores	0%	0%	10%	20%	20%	50%
Expertos	0%	0%	10%	10%	40%	40%

Gráfico 15. El sistema facilita el trabajo colaborativo

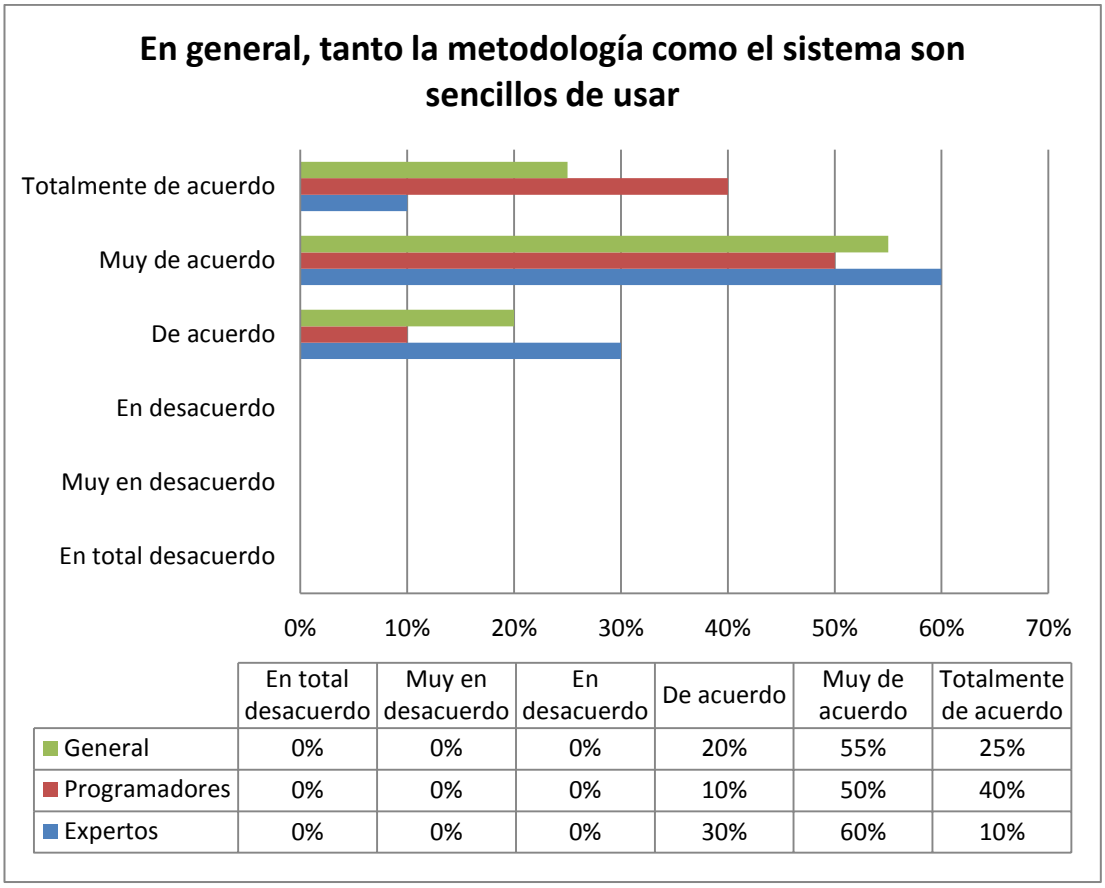
En cuanto a la colaboración entre programador y experto en el dominio que se ha pretendido impulsar y potenciar mediante el uso de la plataforma, el Gráfico 15 muestra que una amplia mayoría está en cierto grado de acuerdo con que el sistema facilita este trabajo colaborativo (90%). Esta opinión es también compartida por programadores y expertos en el dominio, estando los primeros totalmente de acuerdo en un 50% de los casos y los segundos en un 40% de los casos. De esta forma se constata que la plataforma facilita que tanto programadores como expertos en el dominio puedan desempeñar su trabajo de manera conjunta propiciando la colaboración por mediación de la propia plataforma. Por su parte, hay que mencionar que también un 10% de programadores y un 10% de expertos en el dominio están en desacuerdo con dicha afirmación.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	0%	30%	30%	40%
Programadores	0%	0%	0%	30%	20%	50%
Expertos	0%	0%	0%	30%	40%	30%

Gráfico 16. La metodología de desarrollo facilita el trabajo colaborativo

Uno de los pilares sobre los que se ha concebido la metodología de desarrollo es la implicación de los expertos en el dominio en el proceso de desarrollo de sistemas sensibles al contexto, por lo que es de vital importancia medir si la metodología es adecuada para proporcionar un marco colaborativo entre programadores y expertos en el dominio. Así, se puede constatar que en términos generales, el 100% de los participantes está de acuerdo en cierta medida en que la metodología de desarrollo facilita el trabajo colaborativo. Destaca que un 50% de los programadores afirman estar totalmente de acuerdo con esta afirmación, mientras que un 30% de los expertos en el dominio están totalmente de acuerdo. El porcentaje de los programadores es más significativo debido a que la ayuda que los expertos en el dominio prestan a los programadores con su conocimiento en el ámbito de aplicación es de vital importancia para los mismos, ya que estos no tienen porqué tener conocimiento alguno en dichos ámbitos. De esta manera, se percibe con más valor esta colaboración para los programadores que para los expertos en el dominio.



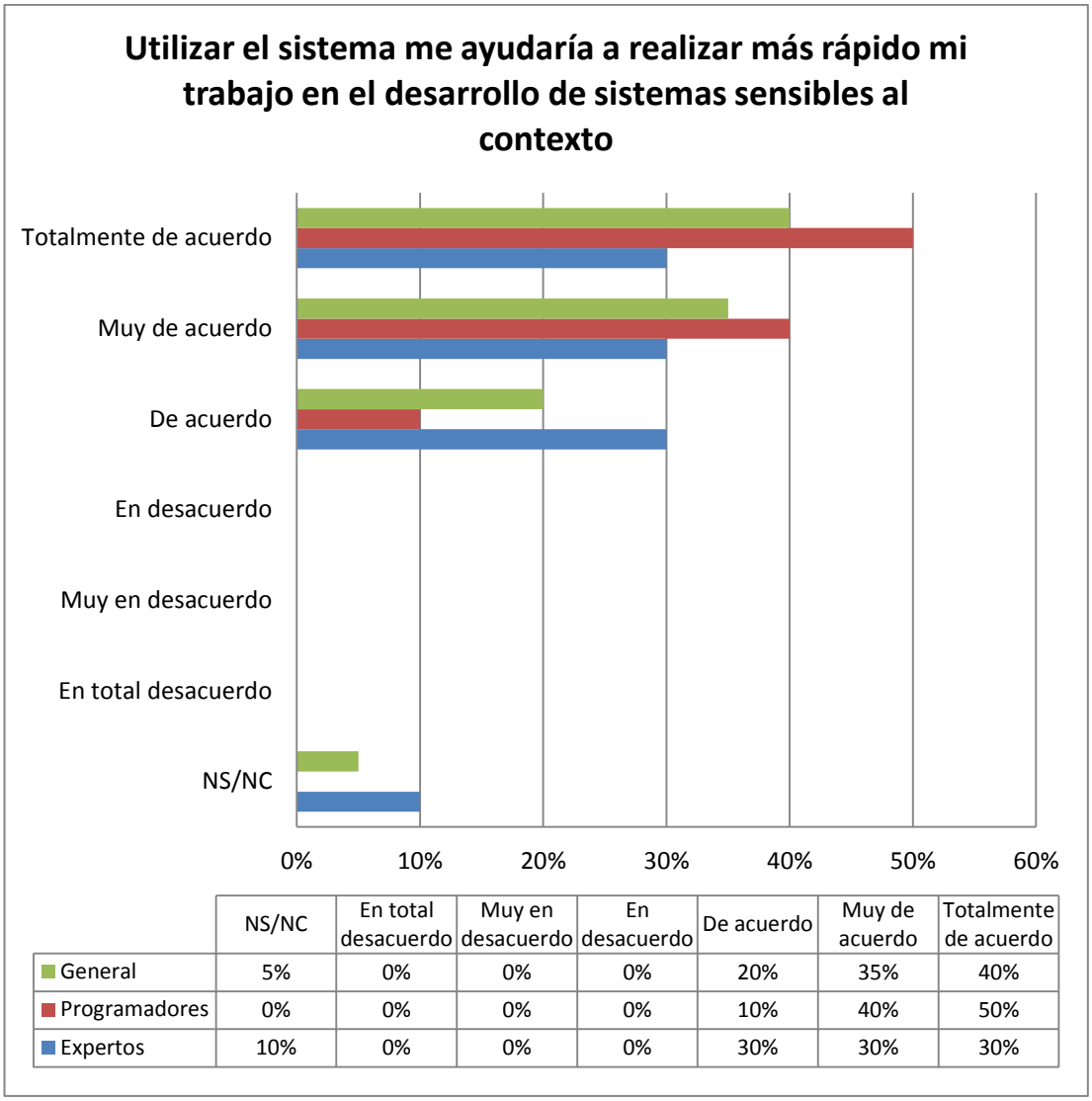
	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	0%	20%	55%	25%
Programadores	0%	0%	0%	10%	50%	40%
Expertos	0%	0%	0%	30%	60%	10%

Gráfico 17. En general, tanto la metodología como el sistema son sencillos de usar

En función de los resultados mostrados en el Gráfico 17, el sistema es sencillo de utilizar para el 100% de los participantes, estando los programadores totalmente de acuerdo con dicha afirmación en un 50% y los expertos en el dominio muy de acuerdo en un 60%. De esta manera, se puede constatar que la plataforma es sencilla de utilizar, siendo los programadores los que más facilidades encuentran en su uso debido al perfil técnico de los mismos.

6.2.2.2. **Utilidad percibida**

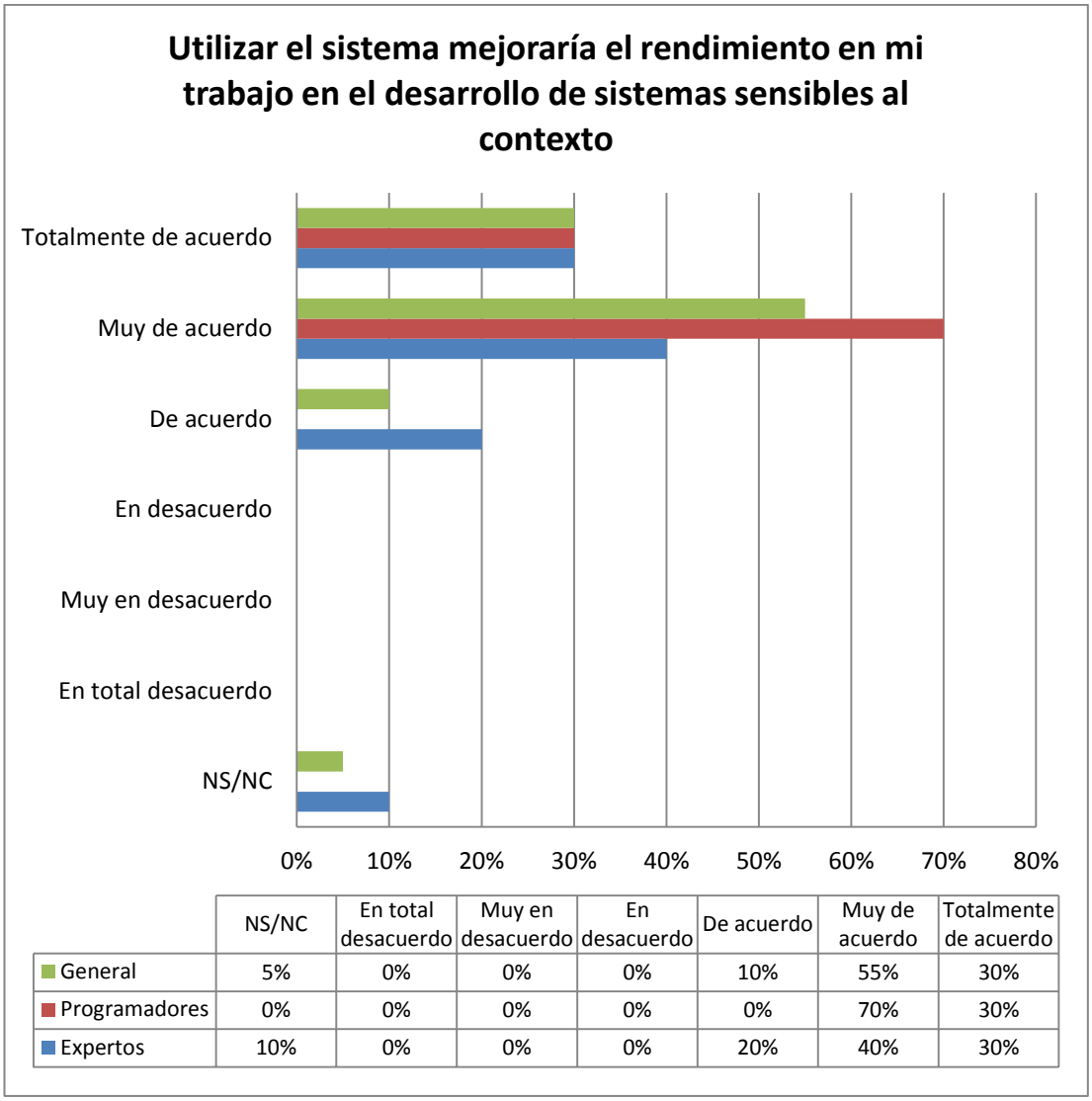
En este segundo bloque, se analizan los resultados relativos a la utilidad percibida tanto de la plataforma como de la metodología de desarrollo. Por cada una de las afirmaciones formuladas en el cuestionario, se muestran los resultados de los programadores en comparación con los resultados de los expertos en el dominio, así como los resultados generales.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	20%	35%	40%
Programadores	0%	0%	0%	0%	10%	40%	50%
Expertos	10%	0%	0%	0%	30%	30%	30%

Gráfico 18. Utilizar el sistema me ayudaría a realizar más rápido mi trabajo en el desarrollo de sistemas sensibles al contexto

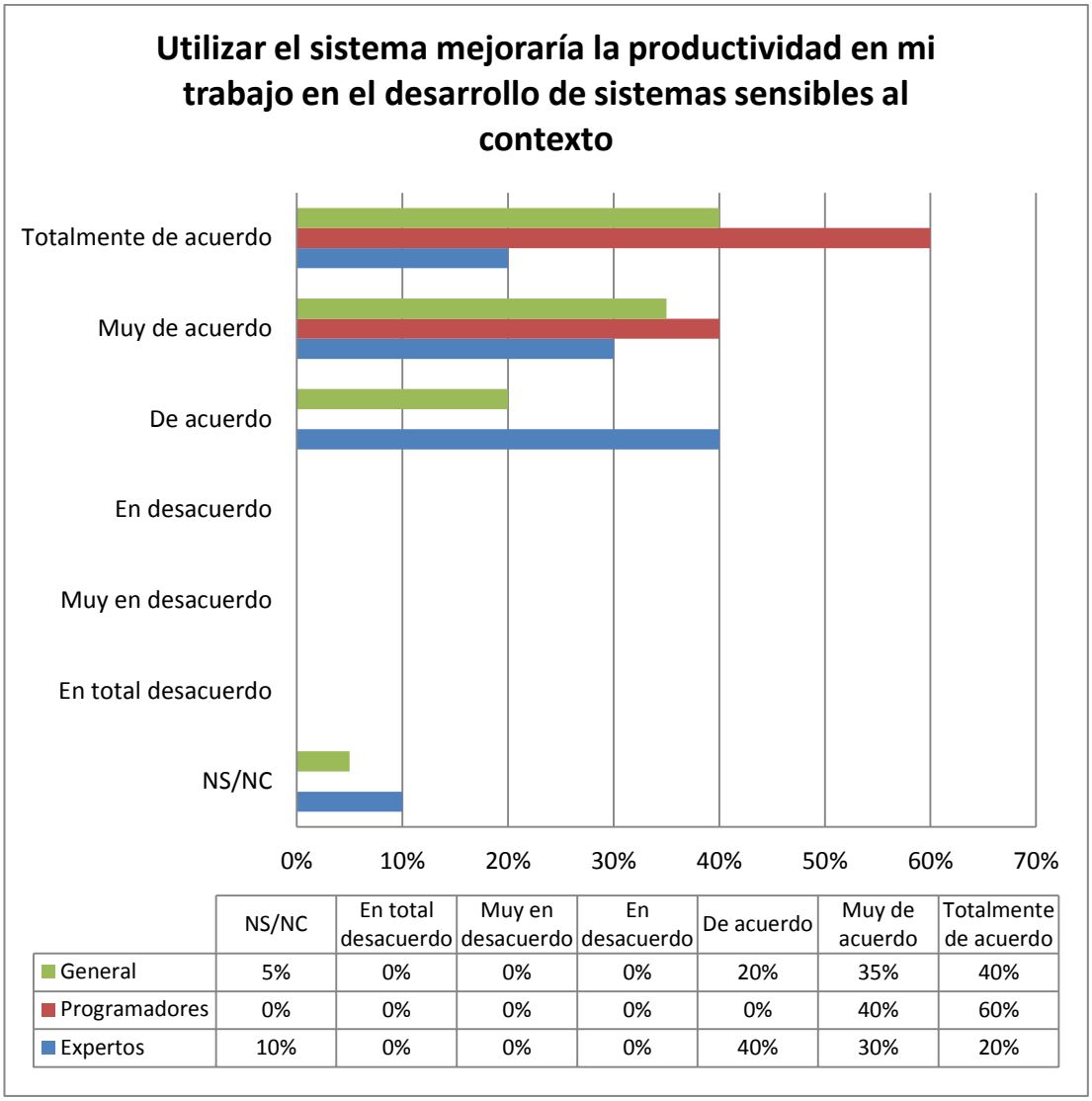
El Gráfico 18 muestra las opiniones de los participantes en relación a la incidencia que tendría la plataforma para realizar más rápido el proceso de desarrollo de sistemas sensibles al contexto. El 95% de los participantes están en cierta medida de acuerdo con dicha afirmación. Así, el 50% de los programadores están totalmente de acuerdo y el 40% de ellos están muy de acuerdo. Esto muestra que los programadores ven que el proceso de desarrollo de sistemas sensibles al contexto puede acelerarse contando con la plataforma. Además, el 30% de los expertos en el dominio también están totalmente de acuerdo. Por lo tanto, este grupo de participantes afirman también que con la plataforma el desarrollo de este tipo de sistemas puede realizarse de una manera más rápida.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	10%	55%	30%
Programadores	0%	0%	0%	0%	0%	70%	30%
Expertos	10%	0%	0%	0%	20%	40%	30%

Gráfico 19. Utilizar el sistema mejoraría el rendimiento en mi trabajo en el desarrollo de sistemas sensibles al contexto

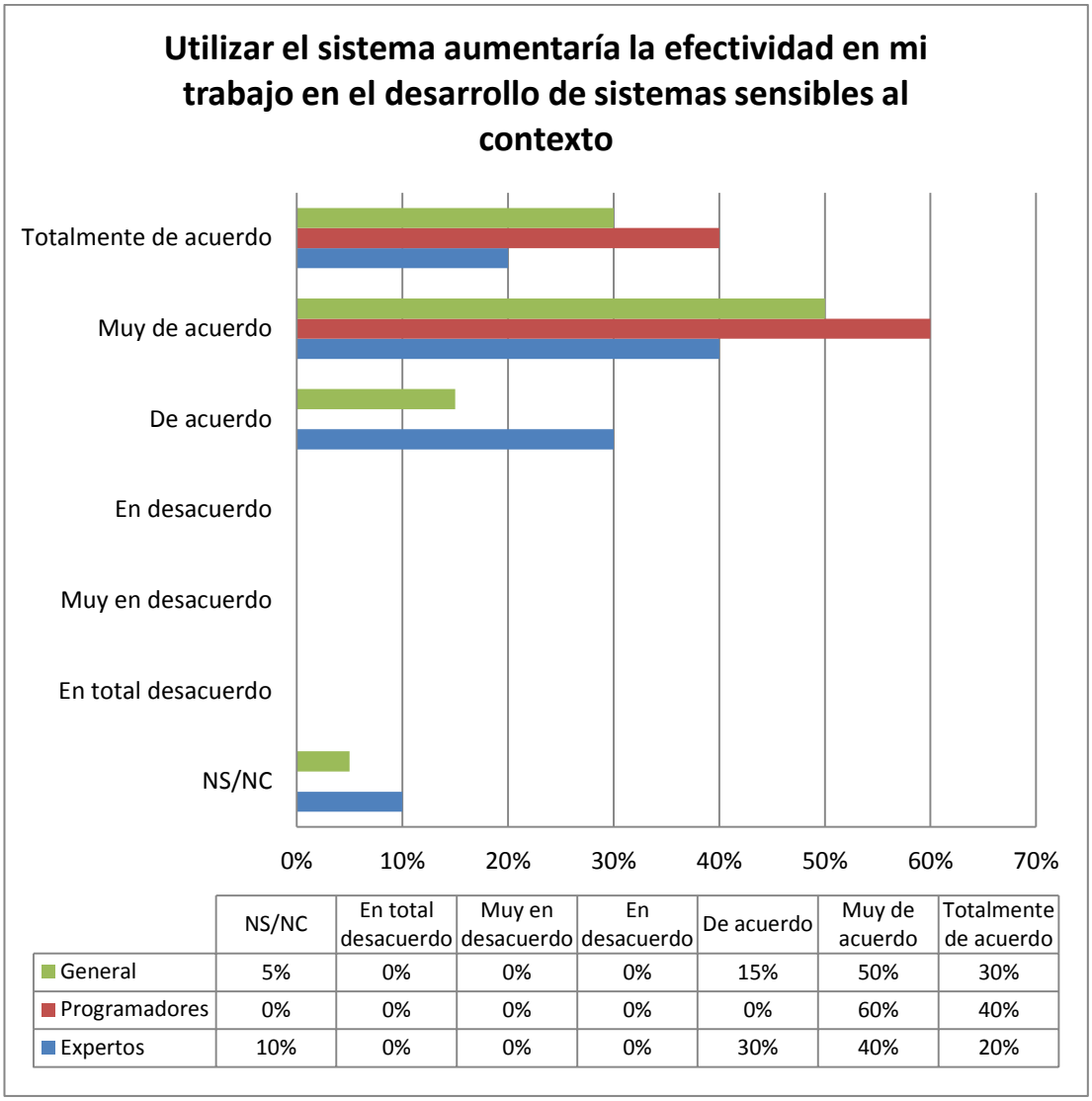
En el Gráfico 19 se muestran los resultados relativos a la afirmación de que utilizando la plataforma, mejoraría el rendimiento en el desarrollo de este tipo de sistemas. Como puede apreciarse, los programadores están de acuerdo en un alto grado con esta afirmación. Concretamente un 30% está totalmente de acuerdo y el 70% restante está muy de acuerdo. Por su parte, los expertos en el dominio están en su mayoría también de acuerdo en cierto grado con esta afirmación (90%). Por lo tanto, los programadores y los expertos en el dominio ven que la plataforma de desarrollo puede mejorar el rendimiento en el desarrollo de sistemas sensibles al contexto.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	20%	35%	40%
Programadores	0%	0%	0%	0%	0%	40%	60%
Expertos	10%	0%	0%	0%	40%	30%	20%

Gráfico 20. Utilizar el sistema mejoraría la productividad en mi trabajo en el desarrollo de sistemas sensibles al contexto

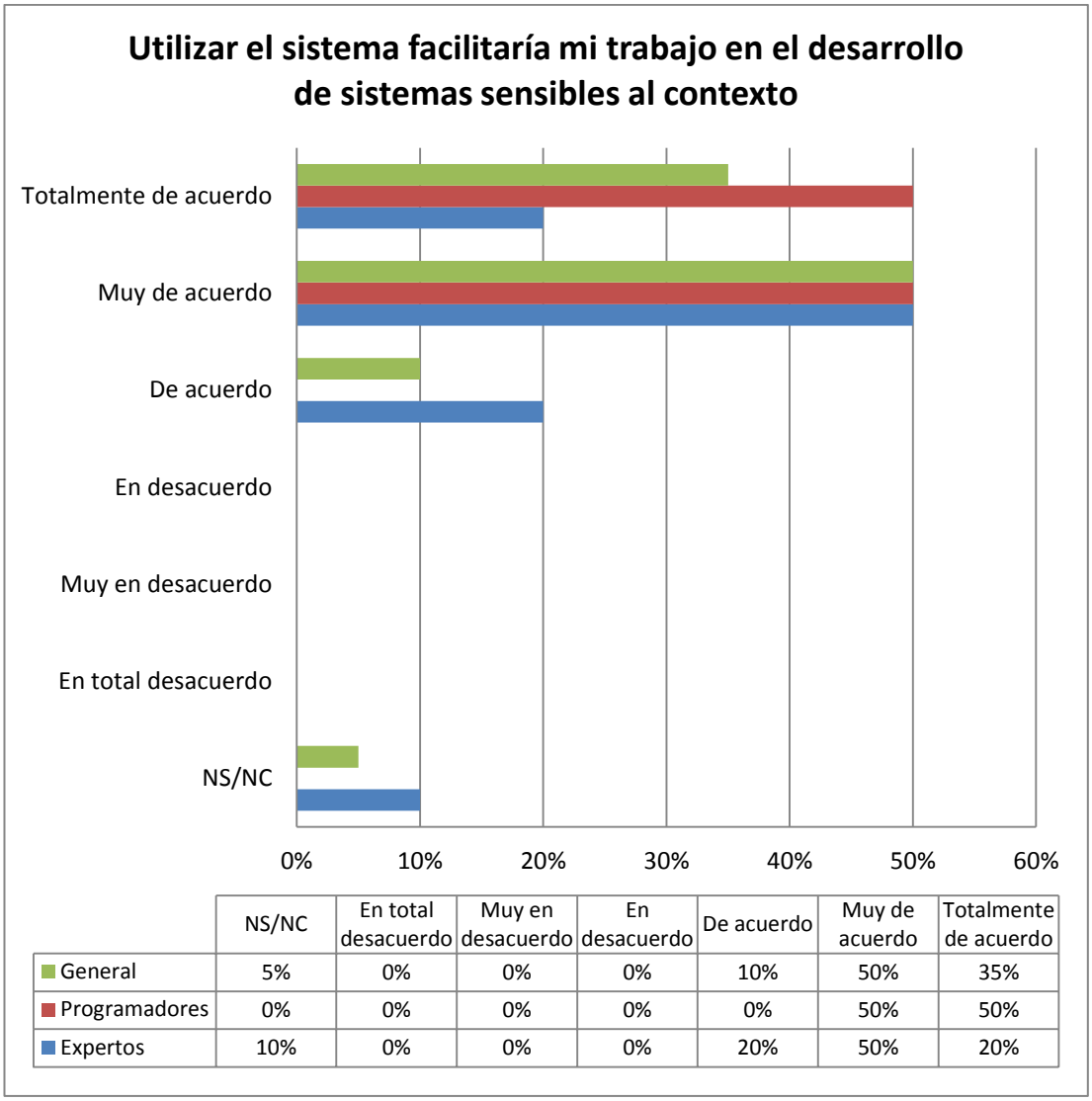
En el Gráfico 20 se muestra que el sistema mejoraría la productividad en el desarrollo de sistemas sensibles al contexto. De esta manera, más de la mitad de los programadores participantes (60%) están totalmente de acuerdo con esta afirmación y un 40% de los mismos están muy de acuerdo. Además, los expertos en el dominio también tienen esa misma apreciación (90%). Por lo tanto, gracias a las funcionalidades que la plataforma ofrece y a la forma de interacción que posibilita sin necesidad de programación, permite que realicen en un menor tiempo las labores de implementación de sistemas sensibles al contexto.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	15%	50%	30%
Programadores	0%	0%	0%	0%	0%	60%	40%
Expertos	10%	0%	0%	0%	30%	40%	20%

Gráfico 21. Utilizar el sistema aumentaría la efectividad en mi trabajo en el desarrollo de sistemas sensibles al contexto

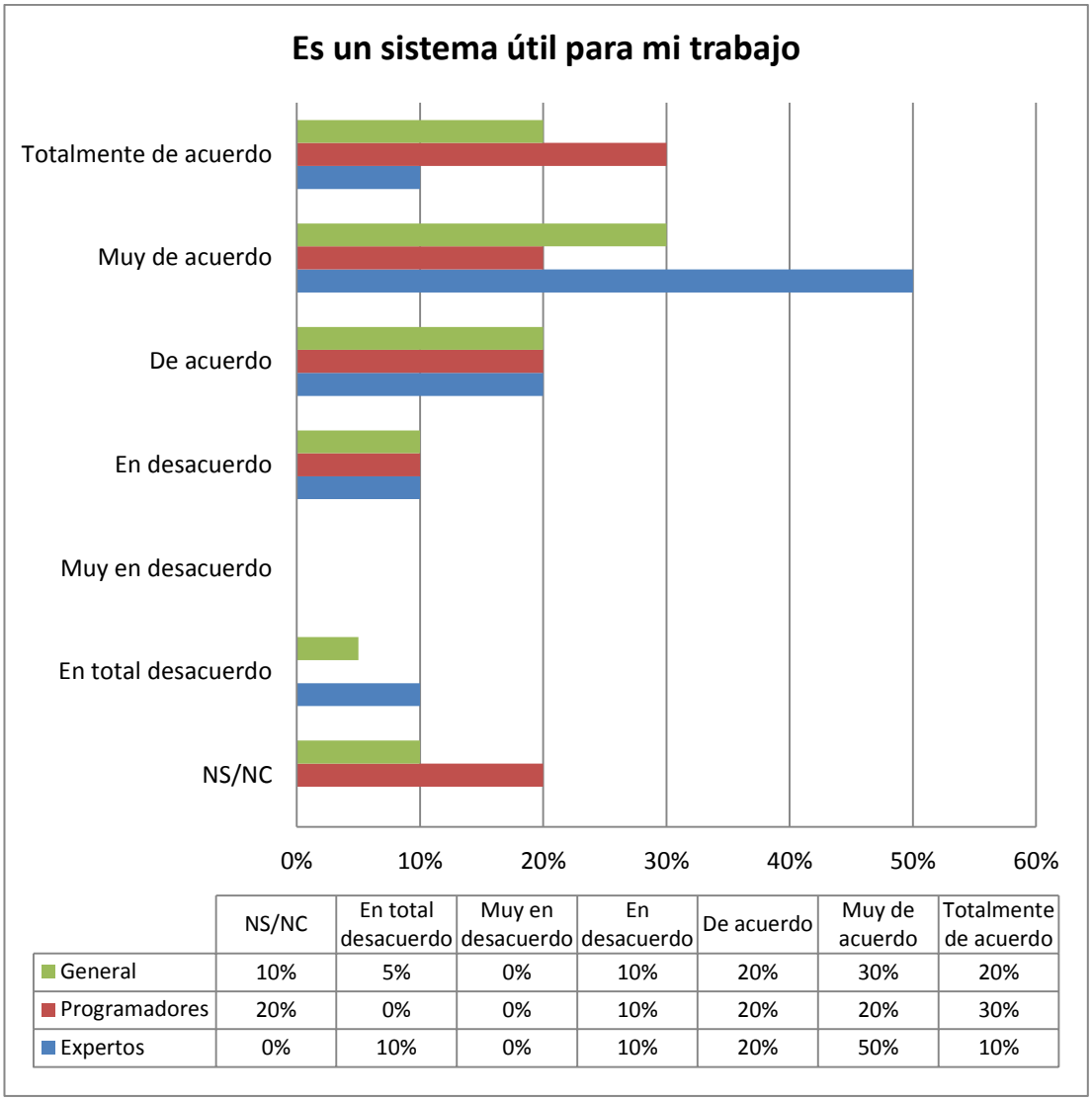
En relación al Gráfico 21, el sistema aumentaría la efectividad en el desarrollo de sistemas sensibles al contexto según el 95% de los participantes. Los programadores son los que más grado de acuerdo muestran ante esta afirmación, con un 40% de los mismos totalmente de acuerdo y un 50% muy de acuerdo. Los resultados relativos a los expertos en el dominio también reflejan que están de acuerdo en cierto grado (90%) con que la plataforma es efectiva para desarrollar este tipo de sistemas. Esto se debe a que las funcionalidades del a plataforma han sido pensadas específicamente para el desarrollo de sistemas sensibles al contexto.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	10%	50%	35%
Programadores	0%	0%	0%	0%	0%	50%	50%
Expertos	10%	0%	0%	0%	20%	50%	20%

Gráfico 22. Utilizar el sistema facilitaría mi trabajo en el desarrollo de sistemas sensibles al contexto

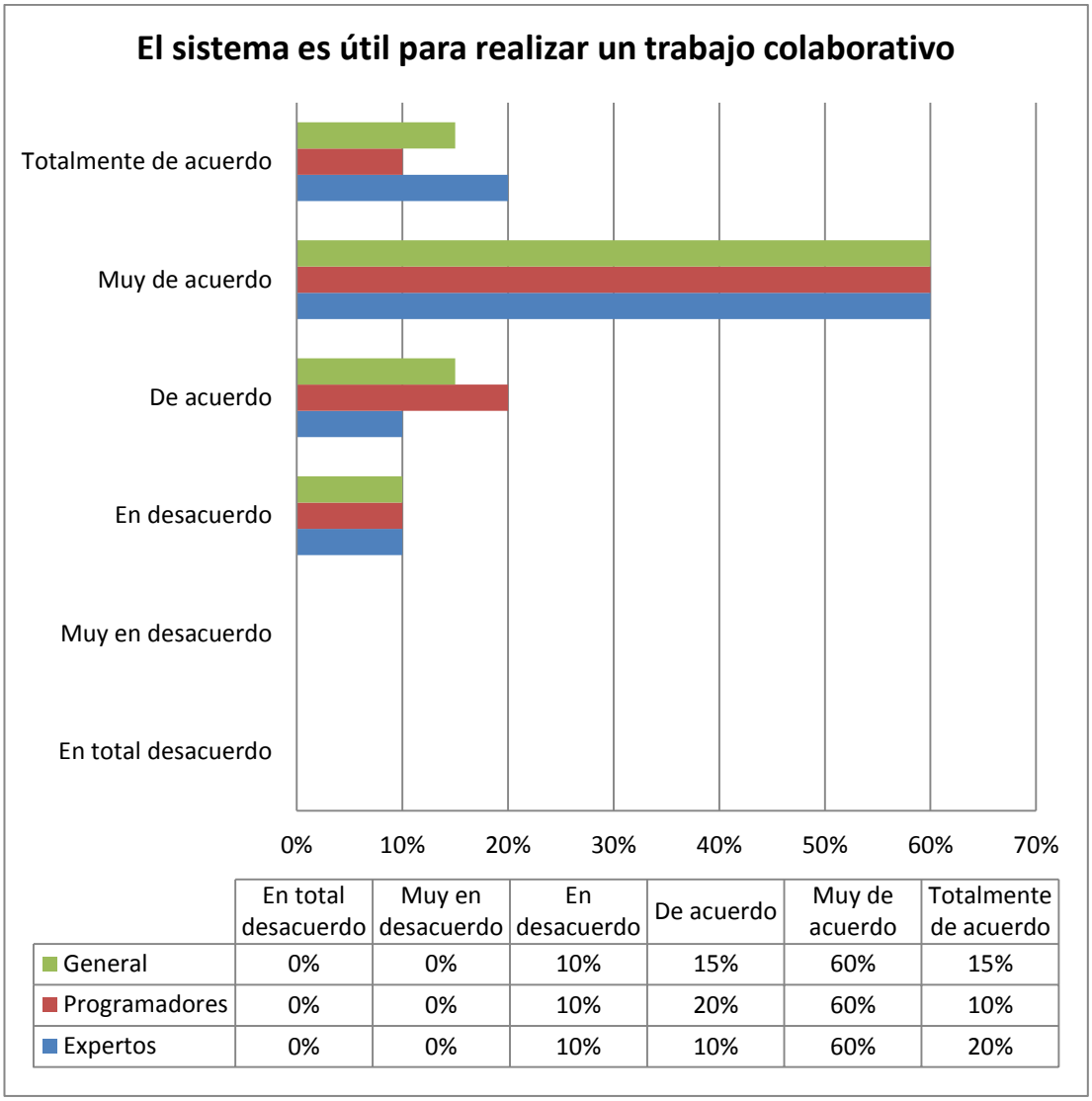
Los programadores están totalmente de acuerdo (50%) o muy de acuerdo (50%) en que la plataforma facilitaría el trabajo en el desarrollo de sistemas sensibles al contexto. Por su parte, también un 70% de los expertos en el dominio están de acuerdo, estando muy de acuerdo un 50% de los participantes de este tipo. De esta manera, tanto los programadores como los expertos en el dominio ven que la plataforma facilita labores de implementación de este tipo de sistemas, por lo que las funcionalidades que ofrece son adecuadas para simplificar el desarrollo.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	10%	5%	0%	10%	20%	30%	20%
Programadores	20%	0%	0%	10%	20%	20%	30%
Expertos	0%	10%	0%	10%	20%	50%	10%

Gráfico 23. Es un sistema útil para mi trabajo

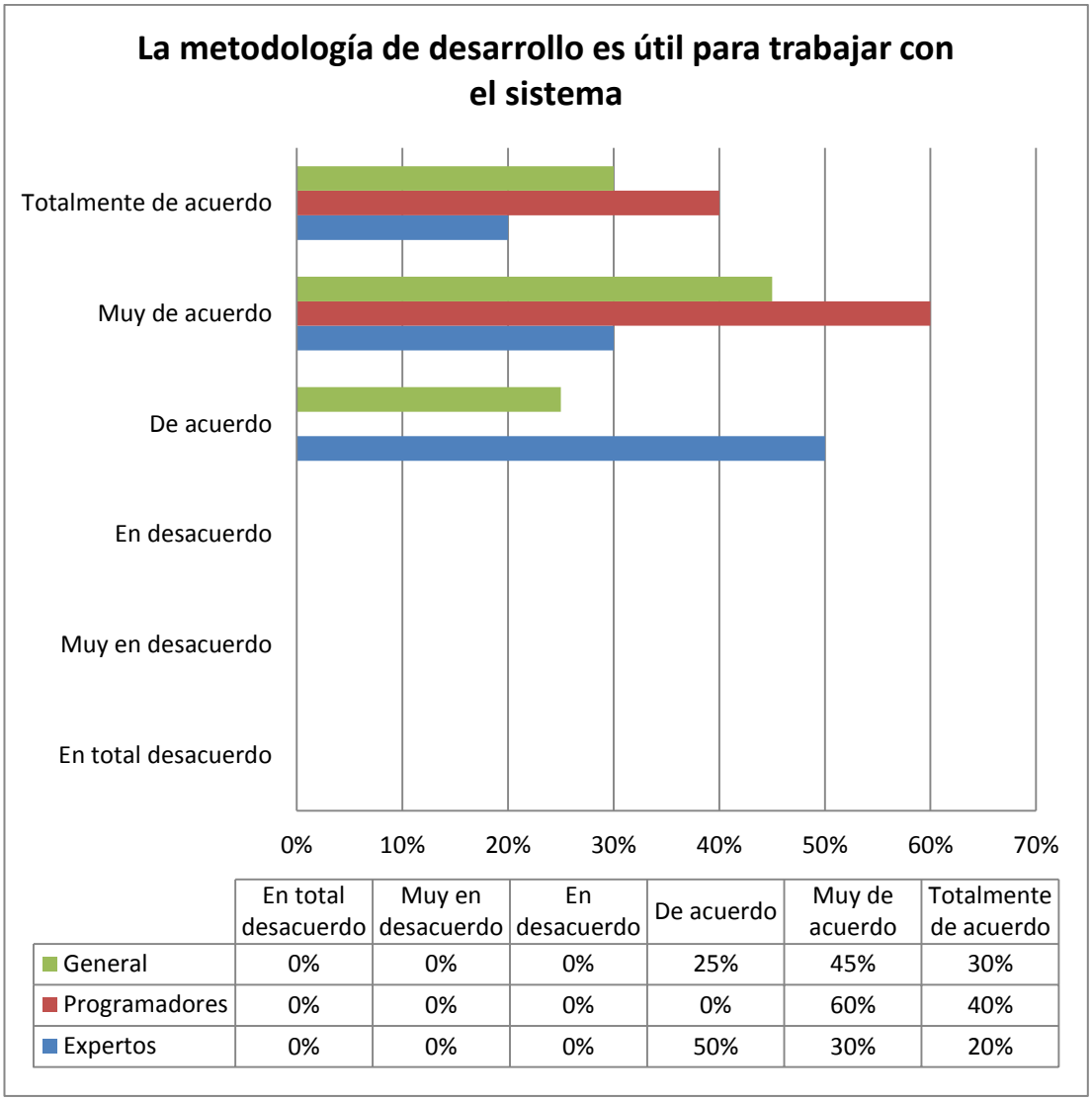
En cuanto a la utilidad de la plataforma para el trabajo desempeñado por los participantes, el 70% de los mismos están de acuerdo en cierta medida que es un sistema útil para poder utilizarlo en su trabajo. Sin embargo existe un porcentaje de participantes que no están de acuerdo con esta afirmación. Tanto los expertos en el dominio como los programadores están en desacuerdo en un 10% de los casos. Además, un 10% de los expertos en el dominio están en total desacuerdo. Cabe destacar también que un 20% de los programadores no han tenido en cuenta esta afirmación. Estos resultados se deben principalmente a que las labores desempeñadas en el trabajo de los participantes no tienen necesariamente relación con la implementación de este tipo de sistemas tan específicos, como son los sistemas sensibles al contexto.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	10%	15%	60%	15%
Programadores	0%	0%	10%	20%	60%	10%
Expertos	0%	0%	10%	10%	60%	20%

Gráfico 24. El sistema es útil para realizar un trabajo colaborativo

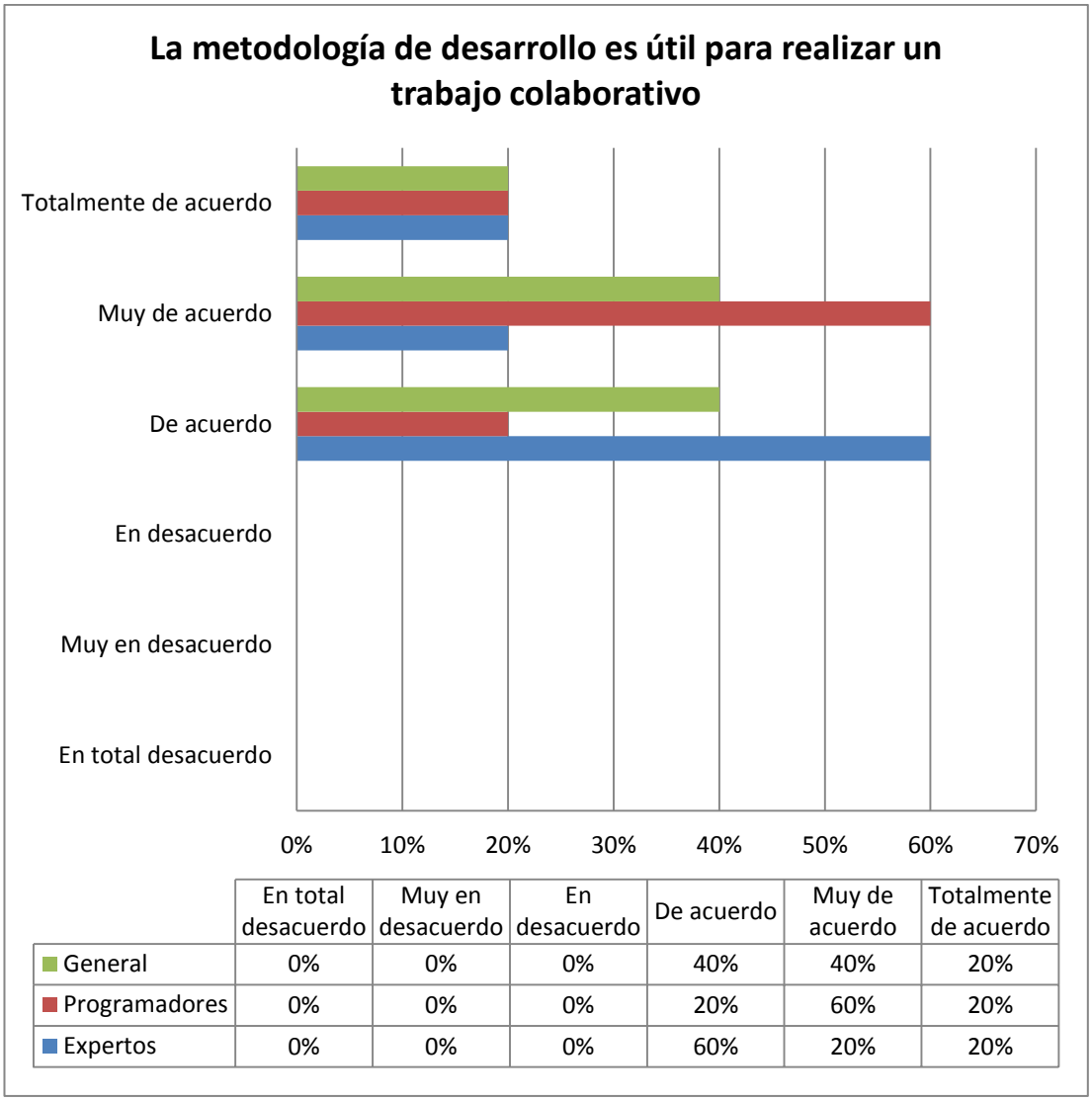
En relación al Gráfico 24, la mayoría de participantes está muy de acuerdo (60%) con que la plataforma es útil para realizar un trabajo colaborativo entre programadores y expertos en el dominio. Además, un 10% de programadores y un 20% de expertos en el dominio están totalmente de acuerdo. Existen también opiniones adversas. Un 10% de los programadores y un 10% de los expertos en el dominio están en desacuerdo.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	0%	25%	45%	30%
Programadores	0%	0%	0%	0%	60%	40%
Expertos	0%	0%	0%	50%	30%	20%

Gráfico 25. La metodología de desarrollo es útil para trabajar con el sistema

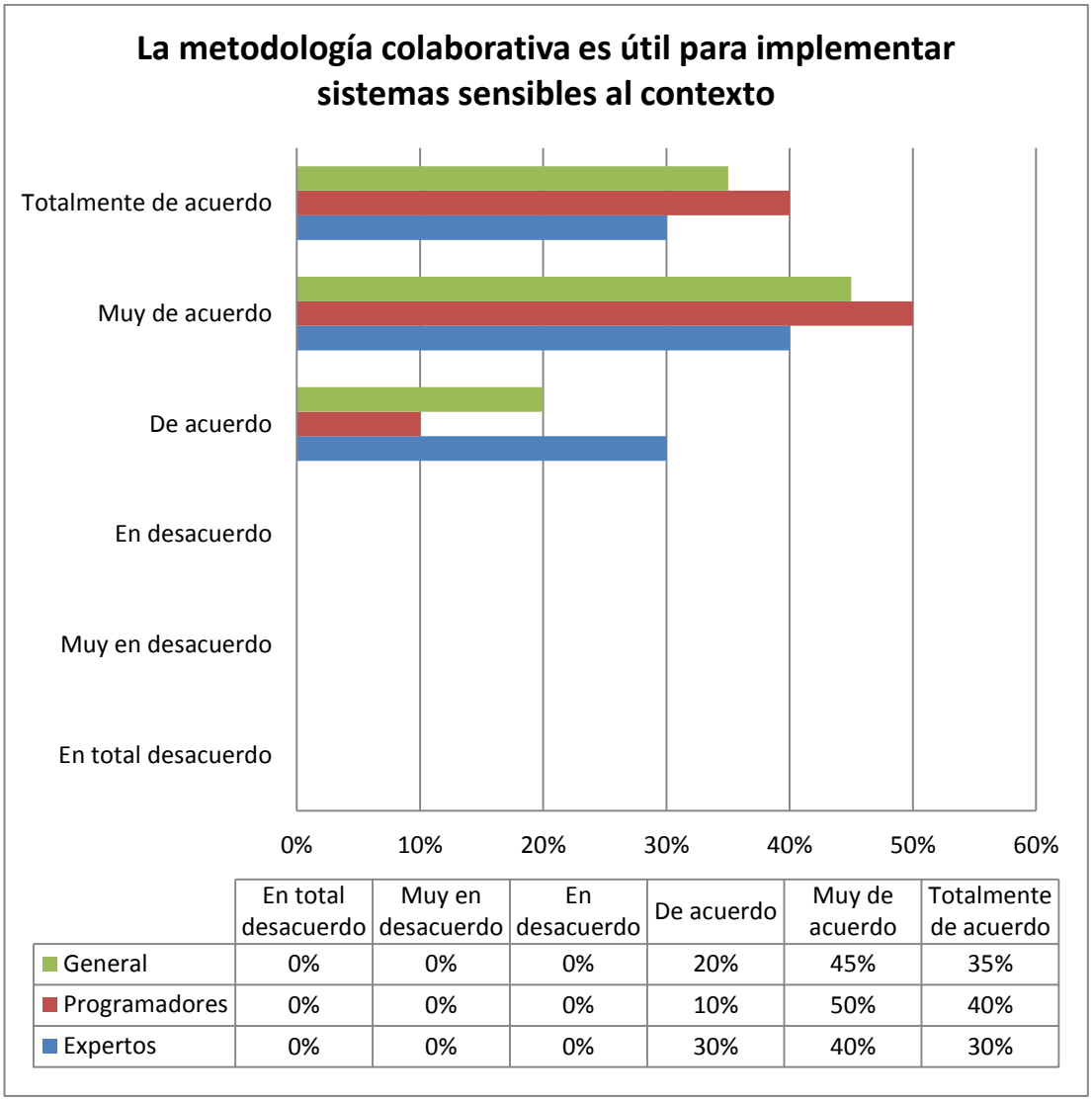
En cuanto a la utilidad de la metodología de desarrollo para trabajar con la plataforma, el 100% de los participantes está en cierta medida de acuerdo. Así, los programadores están muy de acuerdo en el 60% de los casos y totalmente de acuerdo en el 40% de los casos. Por su parte, los expertos en el dominio están de acuerdo en un 50% de los casos, muy de acuerdo en un 30% de los casos y totalmente de acuerdo en el 20% de los casos. De esta manera se aprecia que la metodología resulta un instrumento útil para poder trabajar con la plataforma. El mayor grado de acuerdo de los programadores indica que ellos ven en mayor medida los beneficios de utilizar la metodología para trabajar con la plataforma ya que posibilita la participación de los expertos en el dominio en el propio proceso.



	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	0%	40%	40%	20%
Programadores	0%	0%	0%	20%	60%	20%
Expertos	0%	0%	0%	60%	20%	20%

Gráfico 26. La metodología de desarrollo es útil para realizar un trabajo colaborativo

La metodología de desarrollo es percibida como útil para realizar un trabajo colaborativo por el 100% de los participantes. En este caso también los programadores están de acuerdo en mayor grado, estando muy de acuerdo en un 60% de los casos. Por su parte, los expertos en el dominio están de acuerdo en el 60% de los casos, por lo que su percepción de utilidad es ligeramente inferior a la de los programadores participantes. En general, todos los participantes perciben la metodología como un instrumento útil y válido para realizar un trabajo colaborativo en la definición de situaciones y posterior configuración de las mismas mediante la plataforma.



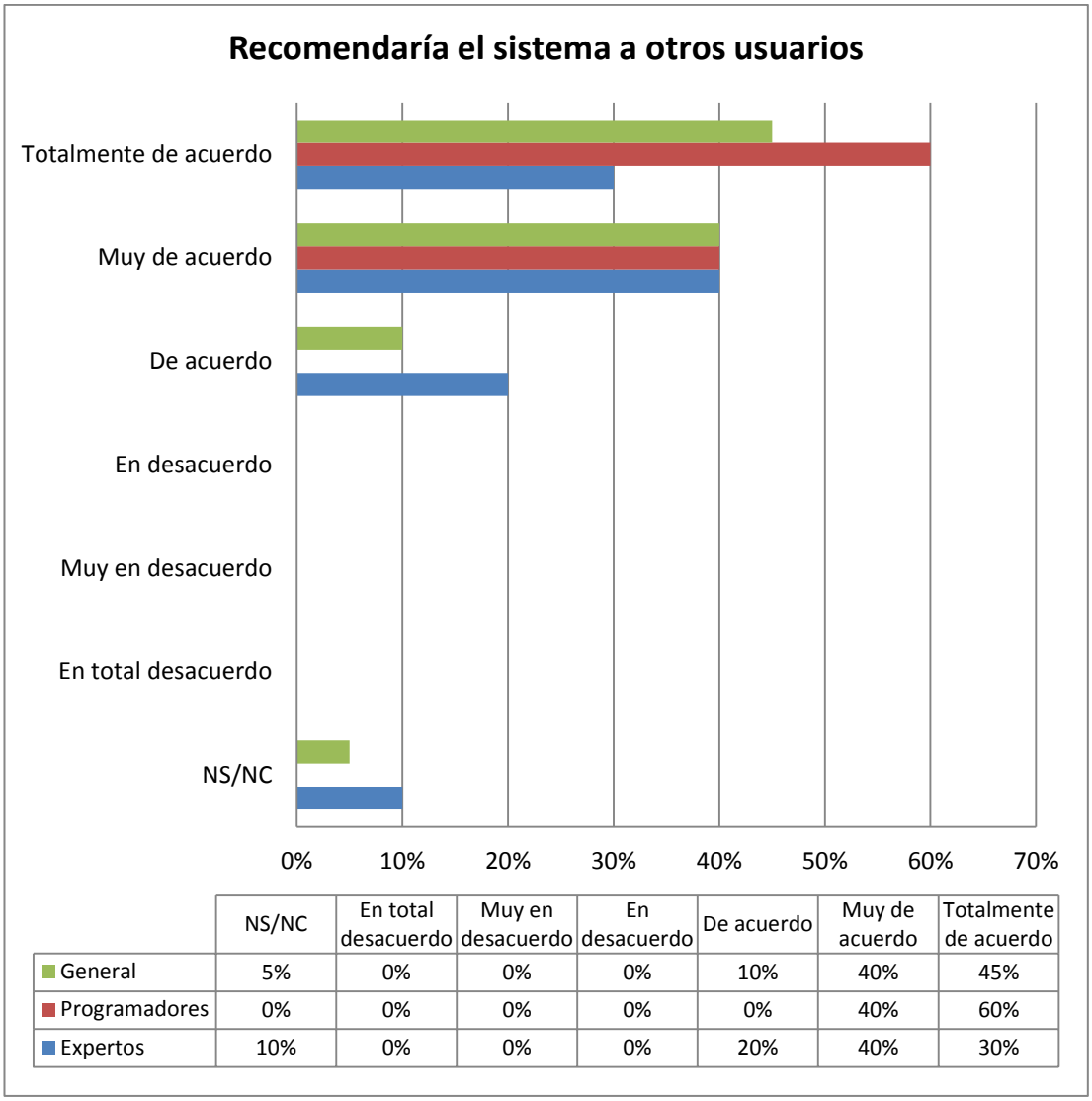
	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	0%	0%	0%	20%	45%	35%
Programadores	0%	0%	0%	10%	50%	40%
Expertos	0%	0%	0%	30%	40%	30%

Gráfico 27. La metodología colaborativa es útil para implementar sistemas sensibles al contexto

Además de ser útil para fomentar la colaboración entre programador y experto en el dominio, el 100% de los participantes está en cierto grado de acuerdo en que la metodología resulta útil para ser aplicada en la implementación de sistemas sensibles al contexto. Así, un 40% de programadores está totalmente de acuerdo y un 30% de los expertos en el dominio está también totalmente de acuerdo con dicha afirmación. En esta ocasión también los programadores están de acuerdo en mayor grado que los expertos en el dominio en cuanto a la utilidad de la metodología en la implementación de sistemas sensibles al contexto.

6.2.2.3. **Intención de conducta**

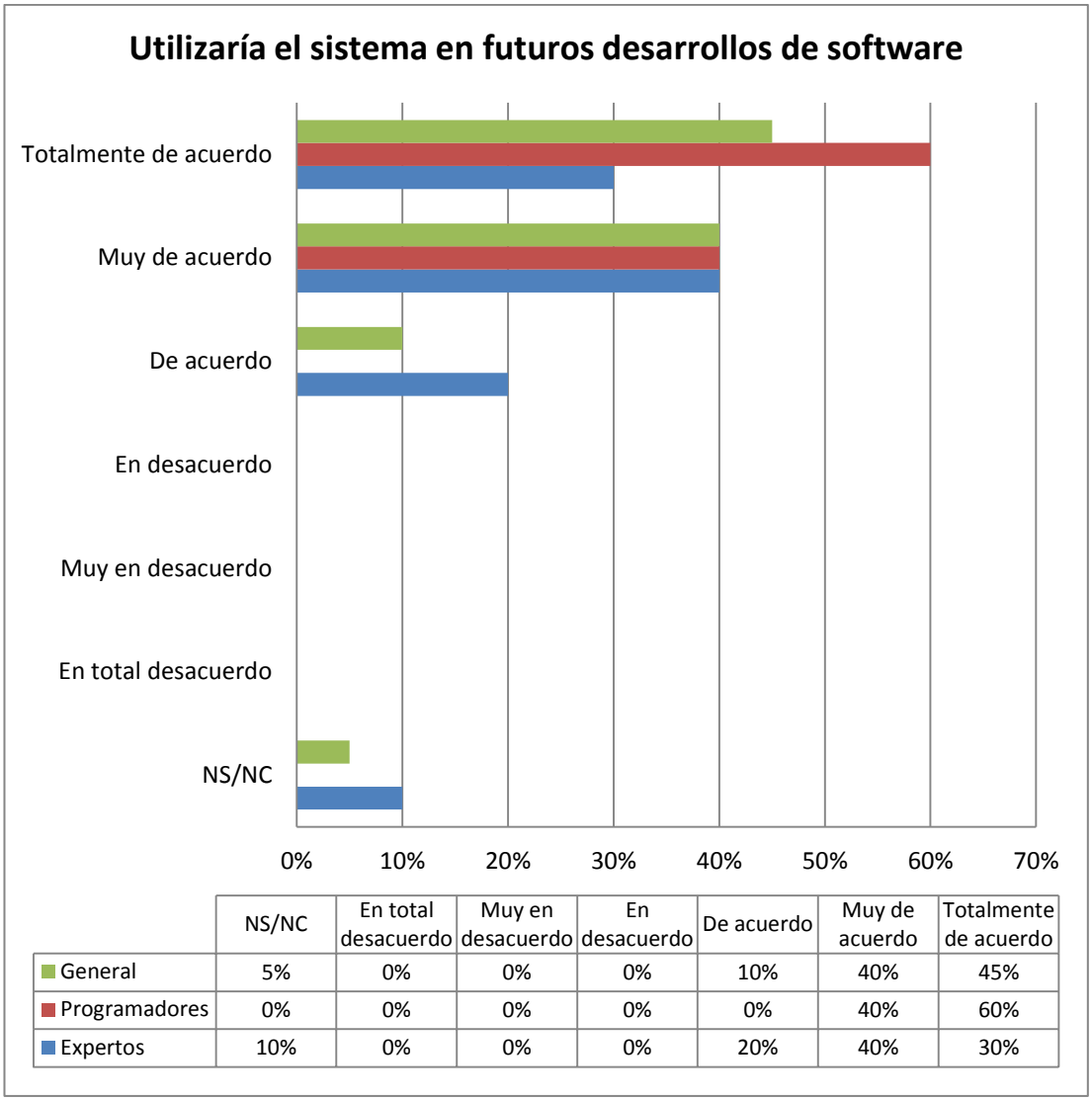
En esta sección se analizan los resultados relativos a la intención de conducta de los participantes en relación a la plataforma y la metodología.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	10%	40%	45%
Programadores	0%	0%	0%	0%	0%	40%	60%
Expertos	10%	0%	0%	0%	20%	40%	30%

Gráfico 28. Recomendaría el sistema a otros usuarios

En función del Gráfico 28, la gran mayoría de los participantes (95%) recomendaría el sistema a otros usuarios. Concretamente, un 60% de los programadores y un 30% de los expertos en el dominio están totalmente de acuerdo con esta afirmación. En esta ocasión también son los programadores los que mayor grado de acuerdo tienen en relación a la afirmación planteada, por lo que son los más dispuestos a recomendar el sistema a otros programadores.



	NS/NC	En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo
General	5%	0%	0%	0%	10%	40%	45%
Programadores	0%	0%	0%	0%	0%	40%	60%
Expertos	10%	0%	0%	0%	20%	40%	30%

Gráfico 29. Utilizaría el sistema en futuros desarrollos de software

En cuanto a la utilización del sistema en futuros desarrollos de aplicaciones que utilicen información de contexto, un 95% de los participantes está en cierto grado de acuerdo con que utilizarían la plataforma. Así, los programadores en su mayoría están totalmente de acuerdo con un 60% de los participantes. Además, los expertos en el dominio son también partidarios de participar en el desarrollo de este tipo de sistemas utilizando la plataforma, estando totalmente de acuerdo con la afirmación en un 30% de los casos.

Además, el 25% de los participantes afirman que pagarían por utilizar la plataforma de desarrollo, pagando una media de 460 euros. Dos de los participantes apuntaron que pondrían un precio por suscripción dependiendo de los servicios requeridos y el uso de los mismos por parte de los usuarios.

6.2.2.4. **Observaciones**

Durante el transcurso de la prueba, un observador apuntó todo aquello que pudiera ser relevante para su posterior evaluación. A continuación se presentan las observaciones más notables encontradas en las pruebas.

- Se constata que la ficha Excel que tiene que ser rellenada entre el programador y el experto en el dominio por cada una de las situaciones identificadas (Sección 4.1.1.3), fomenta el diálogo entre ambos. Se promueve así la colaboración entre los participantes y se establece un diálogo fluido en la fase de parametrización de cada una de las situaciones identificadas.

- El programador apoya al experto en el dominio en la fase de configuración de la plataforma en aquellas cuestiones que este último no acaba de entender bien. Por lo tanto, es el programador el que mejor y más rápido asimila los conceptos utilizados en la configuración y uso de la herramienta, pudiendo así guiar al propio experto en el dominio en la fase de configuración. Uno de los puntos en los que los expertos en el dominio más necesitaron el apoyo de los programadores, fue en la identificación y configuración de las entidades que conforman el modelo de datos de contexto (Sección 5.6).

- El manual de usuario proporcionado a los participantes fue de utilidad para los mismos ya que se consultó en diversas ocasiones. La sección más consultada del manual fue aquella en la que se explica un ejemplo de especificación de situación (Sección 4.2.1). Así, programadores y expertos en el dominio tomaron como guía de referencia dicha situación explicada en el manual para poder parametrizar las situaciones que tenían que configurar en el transcurso de la prueba.

- Un punto de confusión general tanto para programadores como para expertos en el dominio fue la identificación de las salidas a ser generadas por la plataforma (Sección 5.8.5). Así, perciben las salidas como la información que el usuario del servicio final recibirá. Esta percepción es errónea, ya que las salidas se utilizan para ser enviadas a un servicio en el que el programador pueda utilizarlas con el fin de programar los comportamientos requeridos en el sistema final. Así, es interesante que las especificaciones de

las salidas a ser generadas por la plataforma una vez sea detectada alguna de las situaciones requeridas, sean realizadas por los programadores, ya que serán ellos los que posteriormente tengan que utilizarlas para implementar los comportamientos requeridos.

- En diversas ocasiones, fue el experto en el dominio el que configuró la plataforma para identificar alguna de las situaciones especificadas en la prueba. Concretamente, 12 de las 40 situaciones que fueron tratadas en el total de pruebas realizadas, fueron configuradas íntegramente por expertos en el dominio. Las 12 situaciones configuradas correspondieron a las situaciones S2 y S3. La S4 fue siempre configurada por un programador, ya que era la que mayor complejidad tenía. Aun así, se demuestra que los expertos en el dominio, una vez se han familiarizado con la plataforma, son capaces de valerse por sí mismos para configurar una situación.

6.2.2.5. **Comentarios**

En la parte final del cuestionario se estableció un campo para que los participantes pudieran realizar sus comentarios por escrito referentes al transcurso de la prueba, la metodología y la plataforma de desarrollo. A continuación se recogen los comentarios realizados por algunos de los participantes, clasificados en comentarios realizados por programadores y comentarios realizados por expertos en el dominio.

- Programadores
    - "Me parece un sistema muy útil y sencillo de usar por cualquier persona" (Usuario 3).
    - "Herramienta muy útil una vez se entiende su manejo. Fácil de construir aplicaciones complejas ya que abstrae la información de bajo nivel de sensores. El punto más complicado de su uso es el conceptualizar lo que se quiere hacer, necesita práctica. Posibilidades infinitas" (Usuario 7).
    - "La plataforma sería más colaborativa si distintos usuarios pudieran compartir los distintos objetos creados (áreas, proveedores, reglas, etc.) y reutilizarlos. También debería existir un control de versiones sobre los diferentes objetos creados" (Usuario 9).

- Expertos en el dominio
    - "Se echa en falta una explicación de los errores de validación al rellenar los campos de texto" (Usuario 2).
    - "Teniendo nulos conocimientos de programación creo que no me costaría mucho aprender a usarlo" (Usuario 4).
    - "Al principio es difícil hacerse con todos los conceptos presentados (entidades, reglas, proveedores, etc.)" (Usuario 8).

En general, los comentarios realizados por los participantes son a nivel funcional, requiriendo por parte de los programadores alguna funcionalidad extra y por los expertos en el dominio, información adicional sobre los errores que se presentan a la hora de configurar los diferentes elementos de la plataforma. Por otra parte, se menciona también la dificultad inicial al utilizar la herramienta, aunque también se hace hincapié en su facilidad de uso y posibilidades ofrecidas una vez se aprende a utilizar.

6.2.2.6. **Tiempos**

Durante la prueba se midieron también los tiempos empleados en la parametrización y configuración de cada una de las situaciones. Así, el tiempo medio empleado por cada una de las parejas en parametrizar y configurar la totalidad de las situaciones utilizando la metodología y la plataforma de desarrollo ha sido de 90 minutos.

La primera situación que los participantes tuvieron que configurar fue en la que más tiempo invirtieron, con una media de 37 minutos. Esto se debe al desconocimiento inicial tanto de la metodología como de las funcionalidades de la plataforma.

En relación a la segunda situación, el tiempo medio empleado en su configuración fue de 20 minutos. Este tiempo es menor que el empleado en la primera situación, aun teniendo esta segunda situación una complejidad equiparable a la primera. Esto indica que una vez que los participantes se hacen con el control de la metodología y de la plataforma, adquieren más soltura en la resolución de las situaciones.

Este hecho queda también reflejado en el tiempo medio empleado en la resolución de la tercera situación, donde los participantes emplearon una media de 6 minutos. En esta ocasión también influye que esta tercera situación es muy similar a la primera y que

gran parte de las configuraciones de la plataforma necesarias para su identificación ya estaban realizadas.

Finalmente, la cuarta y última situación fue configurada por los participantes en un tiempo medio de 26 minutos. Destaca que el tiempo empleado es ligeramente superior al empleado en la resolución de la segunda situación. Esto se debe a que esta cuarta situación es la que mayor complejidad presentaba de las cuatro. Aun así, en comparación con el tiempo empleado en la primera situación, el tiempo medio sigue siendo inferior.

6.2.3. Consideraciones finales

A continuación, se describen los resultados más significativos recogidos de los cuestionarios realizados por cada uno de los participantes, de las observaciones y comentarios realizados por los mismos, así como de los tiempos analizados.

En cuanto a la facilidad de uso percibida, en términos generales, los participantes están de acuerdo en que la plataforma es un sistema fácil de usar. Un dato a destacar es que los programadores están en mayor grado de acuerdo con esta afirmación. Este dato es comprensible debido a su alta familiarización con este tipo de entornos de desarrollo.

Concretamente, en relación al aprendizaje del uso de la plataforma, una amplia mayoría de los participantes (95%) afirman que es un sistema fácil de aprender, resultando más sencillo su aprendizaje para los programadores. En cuanto al aprendizaje de la metodología, los resultados se repiten, siendo los programadores los que más facilidad han encontrado en su aprendizaje. Aun así, los expertos en el dominio consideran también en su amplia mayoría (90%) que les ha resultado fácil aprender a utilizar la metodología. Estos datos evidencian que tanto las bases de la metodología de desarrollo como las funcionalidades básicas de la plataforma mostradas en el curso de aprendizaje[52] son fácilmente asimilables tanto por programadores como por expertos en el dominio, si bien los primeros tienen más facilidad para asimilar los conceptos involucrados.

En relación al manejo de la plataforma, existe una clara diferencia entre programadores y expertos en el dominio. Por lo general, los programadores afirman haber podido hacer lo que querían en cada momento con la plataforma (90%), si bien

[52] https://vimeo.com/contextcloud

hay algunos que afirman no estar de acuerdo con este hecho. Destaca en esta ocasión que un 40% de los expertos en el dominio no están de acuerdo con esta afirmación. Esto se debe a la cantidad de nuevos conceptos y funcionalidades que debían manejar para poder utilizar la plataforma y su falta de costumbre en el manejo de este tipo de entornos. En las observaciones realizadas, se pudo apreciar que los expertos en el dominio requieren de las explicaciones de los programadores para la realización de ciertas tareas, por lo que se evidencia que la colaboración entre ambos resulta imprescindible y de vital importancia, tal y como se plantea inicialmente en el presente trabajo de investigación. Además, se observó que existen ciertos aspectos que son difíciles de asimilar en fases iniciales del uso de la plataforma, como la identificación y configuración de las salidas que tienen que ser generadas por la plataforma o la identificación de las entidades necesarias.

Cabe destacar que la percepción general de los participantes una vez que tienen soltura en el manejo de la plataforma, es que resulta un sistema fácil de manejar, estando un 100% de los programadores y un 90% de los expertos en el dominio de acuerdo en cierto grado. Además, esta evidencia se refuerza con la existencia de un alto porcentaje de participantes que está totalmente de acuerdo en que sería fácil llegar a ser un usuario experto en el manejo de la plataforma, con un 70% de programadores y un 40% de los expertos en el dominio. Lo que es más, en las observaciones realizadas se contabilizó un 30% de las situaciones que fueron configuradas en la plataforma íntegramente por los expertos en el dominio. Por lo tanto, estos resultados evidencian que a medida que los usuarios de la plataforma la utilizan con mayor frecuencia, ganan en seguridad en el manejo de la misma y resulta más sencillo utilizar de manera correcta las funcionalidades ofrecidas por la plataforma, tanto para los programadores como para los expertos en el dominio.

En cuanto al marco colaborativo que se tiene como objetivo entre programadores y expertos en el dominio, destaca que un 90% de programadores y otro 90% de expertos en el dominio están en cierto grado de acuerdo en que la plataforma facilita el trabajo colaborativo. Además, la metodología es también percibida como facilitadora de dicha colaboración de una manera más clara que la propia plataforma, estando la totalidad de participantes de acuerdo con dicha afirmación. Estos datos evidencian que tanto la concepción de la plataforma como de la metodología son acertadas a la hora de promover la colaboración entre programadores y expertos en el dominio para el

desarrollo de sistemas sensibles al contexto. Los que con más valía perciben esta colaboración son los programadores, ya que son los que más grado de acuerdo presentan. Esto se debe a que para los programadores el conocimiento de los expertos en el dominio es de vital importancia en la configuración de la plataforma y en el desarrollo de la solución final, ya que ellos carecen del conocimiento necesario en el dominio de aplicación concreto.

En relación a la utilidad percibida existe un alto grado de acuerdo en que la plataforma y la metodología resultan de utilidad para el desarrollo de sistemas sensibles al contexto. Así, el total de participantes están en cierto grado de acuerdo en que la plataforma ayudaría a realizar el desarrollo de sistemas sensibles al contexto con más rapidez, estando totalmente de acuerdo con esta afirmación un alto porcentaje de programadores (50%) y expertos en el dominio (30%).

Además, casi la totalidad de los participantes (95%) están en cierto grado de acuerdo en que la plataforma mejoraría el rendimiento, la productividad y la efectividad en el desarrollo de este tipo de sistemas. Así, utilizando la plataforma se obtienen buenos y esperados resultados en el desarrollo de sistemas sensibles al contexto, en un menor tiempo de desarrollo e invirtiendo el menor número de recursos posibles.

Por lo general, son los programadores los que mayor grado de acuerdo presentan ante estas tres mejoras, ya que son los que más percepción pueden tener de lo que puede costar implementar un sistema sensible al contexto sin contar con una plataforma como la aquí implementada. En general, un 70% de los participantes perciben que es un sistema útil para su trabajo. Estos datos evidencian que las funcionalidades ofrecidas por la plataforma son las correctas para el desarrollo de este tipo de sistemas y además resultan de gran utilidad.

En cuanto a la utilidad percibida del sistema para realizar un trabajo colaborativo, el 90% de los participantes están en cierto grado de acuerdo con dicha afirmación, habiendo un alto porcentaje de programadores (60%) y expertos en el dominio (60%) que están muy de acuerdo con la misma. En cuanto a la utilidad de la metodología, el 100% de los participantes la perciben como una metodología útil para trabajar con la plataforma.

Además, la totalidad de participantes están de acuerdo en que la metodología es útil para realizar un trabajo colaborativo, siendo una vez más los programadores los que en

mayor medida aprecian esta utilidad. Este hecho se refuerza con las observaciones realizadas durante la prueba, en las que se constata que la fase de requerimientos y parametrización de situaciones mediante la ficha diseñada, fomenta el diálogo entre programador y experto en el dominio. Lo que es más, el total de participantes están en cierto grado de acuerdo en que la metodología es útil para ser aplicada en la implementación de sistemas sensibles al contexto en general. Estos datos evidencian que la metodología es la adecuada para el desarrollo colaborativo de sistemas sensibles al contexto utilizando como herramienta la plataforma desarrollada.

En relación a la intención de conducta, la totalidad de programadores participantes recomendaría el sistema a otros usuarios y lo utilizaría en futuros desarrollos de sistemas sensibles al contexto. Incluso los expertos en el dominio están en cierto grado de acuerdo (90%) en que recomendarían el sistema.

Atendiendo a los tiempos empleados, no existen diferencias significativas entre las diferentes parejas participantes que tomaron parte en la prueba. De esta manera, existen suficientes indicios para establecer que el manejo de la misma es independiente del nivel de conocimientos de programación que tengan los integrantes de las parejas que afrontan un desarrollo por mediación de la metodología y la plataforma. Cabe destacar que fue en la primera situación en la que más tiempo tuvieron que emplear los participantes. Este dato corrobora que una vez que los participantes se hacen con el control de la plataforma les resulta más fácil su configuración, ya que posteriores situaciones fueron configuradas en menor tiempo que la situación inicial.

En líneas generales, la utilidad percibida es más alta que la facilidad de uso. Los programadores son los que mayor grado de acuerdo presentan sobre las afirmaciones realizadas en el cuestionario repartido y son además los que en mayor grado perciben la utilidad de la colaboración con los expertos en el dominio.

Chapter 7 Conclusions

"Let the future tell the truth, and evaluate each one according to his work and accomplishments. The present is theirs; the future, for which I have really worked, is mine."

Nikola Tesla (1856 - 1943). Serbian electrical engineer.

Context information will play a crucial role in the development of next generation software and hardware systems. The personalisation and adaptation of such systems using context data will increase the users' satisfaction providing more usable products and a better user experience. These systems will be able to obtain, process and interpret context information in order to adapt their behaviour and functionalities for the benefit of users.

This dissertation has underlined the pertinence to work further on the definition and the boundaries of the terms *context* and *situations* (Section 2.1), in order to move forward in the adoption of such concepts by the software industry. It has been argued that with a precise definition of these terms, programmers will be able to model context information, and thus, computer systems will be able to manage and process it. Hence, this research work puts forward new definitions of *context* and *situation* that can be used efficiently by programmers in order to model context information (Chapter 2).

This work has evidenced the necessity to improve the development of context-aware systems. On the one hand, the implementation of context-aware systems must be eased by providing software toolkits or frameworks with adapted functionalities. The literature review still reveals some gaps related to these functionalities (Section 2.8). This dissertation presents a web development platform that eases the development of these kinds of systems that fulfils the identified gaps.

On the other hand, it has been shown that the development processes of context-aware systems must be improved as well. As mentioned in Section 2.9, there are no

methodologies that can be specifically applied to the development of context-aware systems. Furthermore, it is also difficult for programmers to identify the relevant situations that the system has to be able to detect in order to adapt its behaviour. This task involves the specification of the needed context information and the desired behaviours of the system once a situation is detected. These parameters are dependent on the application domain and programmers often do not have the needed knowledge about all the targeted application domains. Therefore, the involvement of domain experts in the development stage is needed. This dissertation proposes a development methodology designed for the collaboration among domain experts and programmers, which improves the development process and the final results.

Thus, this dissertation has addressed the limitations found in the reviewed research works, which are deemed to improve the development of context-aware systems in order to create more personalised and adapted systems.

In this chapter, section 7.1 summarizes the most relevant contributions put forward by this dissertation. It first brings back the research questions, the hypothesis and the objectives set in Section 1.2 and describes and analyses the extent to which they have been addressed.

Section 7.2 explains a number of research and scientific implications for the discipline of context-aware computing and more precisely, for the development of context-aware systems.

Both the technical and user evaluations carried out in Chapter 6 have enabled to elucidate areas that deserve further investigation. All of these new research opportunities are specified in Section 7.3.

7.1. Contributions

The present dissertation has described the problems found in context-aware computing and some of the motivations that have activated this research (Section 1.1). These have led to set the research framework for this piece of work in terms of research questions, the hypothesis with which the research work has been validated and finally, the associated goals of this dissertation.

The background presented in Chapter 1 and Chapter 2 has led to the theoretical work presented in Chapter 3. This work has been used in order to guide the two main developments carried out in this work.

- The *Situation-Driven Development* methodology, which is designed to guide the development of context-aware systems through the collaborative involvement of domain experts and programmers (Chapter 4).
- The *Context Cloud* platform, which is a development platform, designed for the detection of situations based on context information (Chapter 5). It is designed for non-technical people, so that programmers and domain experts can collaborate in the development process of context-aware systems guided by the previously mentioned *Situation-Driven Development* methodology.

In a final stage, the collaborative methodology and the development platform set forward have been validated with real users. A total amount of ten pairs of participants composed of a programmer and a tourism domain expert were involved in the evaluation. In addition, empirical observations were carried out to find out the performance of the platform (Chapter 6).

The general research questions (RQ) identified in this dissertation were the following.

- *RQ1*. Can the context information be used in order to personalize systems and services in ubiquitous computing environments?
- *RQ2*. Can context data be managed in order to identify high-level situations using a software toolkit without programming?
- *RQ3*. Is it possible to involve domain experts in the development of context-aware systems?
- *RQ4*. Is it possible to promote the collaboration among domain experts and programmers using a methodology for the development and enhancement of context-aware systems?

In order to provide an answer to these general research questions, the following hypothesis has been put forward.

It is possible to involve domain experts in the development life-cycle of context-aware systems in collaboration with programmers, by providing software toolkits and development methodologies adapted to people that do not have programming skills.

These research questions and hypothesis have led to a number of goals, which are specified next.

- *G1*. To design a collaborative methodology in order to involve domain experts in the development life-cycle of context-aware systems in cooperation with programmers (Chapter 4).
- *G2*. To implement a context-aware software toolkit equally suitable for both programmers and domain experts (Chapter 5).
- *G3*. To provide new definitions of *context* and *situation* (Chapter 3).
- *G4*. To make the development of context-aware systems more universal and democratize the implementation of such systems (Chapter 6).

The literature review in the realm of context-aware computing shows that (i) there is no consensus on a definition for the notions of *context* and *situation*, (ii) there are no specific development methodologies that can be applied for the development of context-aware systems, (iii) the existing context-aware software development toolkits have still some gaps in order to be used in the implementation of such systems and finally, (iv) domain experts are not involved in the development of context-aware systems.

The next sections enumerate the contributions (C#) of this dissertation against the reviewed state of the art and their relation to the different formulated objectives, hypothesis and general research questions.

7.1.1. C1. New definition of the notion of *context*

In recent years, there have been a lot of different context definitions. Some of these definitions consider context as the surroundings of the interaction between the user and the application. Other definitions consider the activity or the task of the user as the main context information for the system. A third group of definitions consider that

context is the needed information to characterize the situation of an entity (Section 2.1.1).

In this research work, context has been considered from a computing perspective, having into account the third group of definitions mentioned before. This way, *context* is considered as *any information that can be obtained and processed by hardware or software systems, in order to identify the situation of an entity and adapt the system's behaviour to that situation*. Extending the definition of context provided by Dey (2001), an *entity* can be *a living being, a place or an object* (Section 3.1). The objective has been to provide a functional definition to be applied in the development of hardware and software solutions that use context information in order to adapt their functionalities. This new approach has been achieved by the consecution of the goal G3 and it has been fundamental for the consecution of the goals G1 and G2 as well.

7.1.2. C2. New definition of the notion of *situation*

Several definitions for the concept of *situation* have been reviewed and analysed as well. All these definitions have a common pattern: they consider context as low-level data, while a situation is considered as high-level data. This way, a situation is dependent on the context information and it can be considered as an abstraction of it (Section 2.1.2).

In the scope of this dissertation, a *situation* is defined as *the state of the relevant current and past context at a certain region in space and a concrete interval in time*. There are two main dimensions that have been taken into account for this new definition. The first one is the notion of *time*. A situation can have temporal boundaries and it can be related to past or current situations (context data). The other one is the notion of *space*, that is, where the situation can be identified. For instance, the situation "cooking" could be detected when the user is in the kitchen (space) and it is time to have lunch (time) (Section 3.1).

This new definition of situation is more operative for the scope of this research work than the ones analysed in the literature review, and its aim is to facilitate the modelling of situations in the development of context-aware systems. This definition has been stated by the consecution of the goal G3 and it has been essential for the consecution of the goals G1 and G2.

7.1.3. C3. New guidelines and requirements to design a context-aware system development methodology

The reviewed software development methodologies fall sort to guide the implementation of context-aware systems due to their general scope. There are some specific methodologies applied to the development of context-aware systems, but they are more programming guidelines than development methodologies (Section 2.9). This dissertation sets forward some requirements, which ideally should to be taken into account when designing a development methodology.

The first requirement establishes that the methodology has to be guided by the definition of situations. This way, situations are the key element around which the methodology has been designed. In order to define a situation, five different characteristics derived from the definition of *situation* have been taken into account. These are related to five different questions that define a situation: the name of the situation (what), the entities that are related to the situation (who), the location where the situation can be detected (where), the time and date range when the situation can be detected (when) and finally, the needed context data in order to detect the situation (how) (Oh et al., 2006).

The second requirement establishes that the methodology has to be designed in order to involve domain experts in the development process in collaboration with programmers. Domain experts can better identify and define the needed situations and they can validate the development process from the beginning (Section 3.2).

The mentioned requirements have been considered in order to design the proposed methodology (Chapter 4) and they can also be used as the guidelines to design new development methodologies in order to implement context-aware systems. This contribution is a consequence of the achievement of the goals G1 and G4, and it has been based in the work carried out to achieve the goal G3.

7.1.4. C4. An improved architecture and new requirements to develop a context-aware software toolkit

There are several software toolkits that have been proposed in order to ease the development of context-aware systems. The reviewed toolkits did not provide all the needed functionalities in order to be used in a context-aware system development (Section 2.8.2). This dissertation has identified the requirements over the common

architecture of such frameworks in order to improve the development of context-aware systems.

This work proposes a software architecture based on the common layers identified in the state of the art: sensor layer, data gathering layer, data processing layer, data management layer and application layer (Section 2.5). Even so, some advances in complementary scientific and technological fields have emerged recently to improve the architecture of these kinds of toolkits and adapt them to the current technological trends. This way, the architecture is enhanced using some of the principles stated in the Web of Things, End-User Programming and Cloud Computing paradigms (Section 3.3.1).

The Web of Things principles have been applied to the sensor layer. This way, sensors must have a web end-point in order to get data from them in a RESTful way. In this manner, these principles provide a higher level of abstraction over the drivers and low-level mechanisms that context data sources expose in order to get data from them.

The End-User Programming paradigm has been considered in order to involve domain experts in the development process in collaboration with programmers. The users of these toolkits must have the knowledge about what is happening behind the scenes and they must have the control over the managed context data without having programming skills (Section 3.3.7).

Cloud Computing paradigm has been applied to the context data gathering and management layers of the architecture. This way, the mentioned layers can be deployed on an external hardware infrastructure, e.g. a data centre. This facilitates the management, reliance, scalability and maintenance of the platform. Furthermore, the developed platform provides the users with a web front-end in order to allow the management of context data at any time and place with any connected device (Section 3.3.8). Hence, domain experts can easily access and modify all the configurations in real time without having to install any development environment.

This research work has considered context-aware systems as reactive systems that adapt their behaviour to the detected situations of the involved entities. This way, the development platform has been designed in order to generate high-level outputs that may directly be exploited by the context-aware applications to be developed in order to adapt their behaviour. There are several context processing patterns that have been

identified in the literature review (Section 2.4), but they are not applicable to a scenario where domain experts are involved in the development process. In order to solve this gap, a reactive pattern has been proposed, where context data is obtained or received by the development platform and it produces several outputs with information about the identified situations. Also, the generated outputs can be used as high-level inputs. The outputs (situations) are modeled using logical rules that domain experts can configure based on their knowledge on the application domain.

In addition, there are other requirements that are related to the context data model (Section 2.6) that have been taken into account in order to store and manage the received context data. This way, the used context data model allows the representation of heterogeneous data, based on the relationships among context entities. Also, reasoning engines can be applied over the specified model. These engines are able to manage spatial and temporal reasoning as well (Section 3.3.3). This way, high-level information (situations) can be detected. Finally, the defined data model is flexible enough in order to be extended at runtime. In this way, the highly dynamic requirements of these kinds of systems can be supported (Section 3.3.2).

This dissertation has considered additional requirements of the architecture related to the management of context data and more precisely, to the context data life-cycle management process. This process involves several tasks. Data has to be transformed into the defined data model, the instances of the model have to be inserted or updated in the knowledge base and data coming from different sources have to be aggregated if they are related to the same entity instance. This management can be quite repetitive so that mechanisms that provide automations have been implemented. Also, a garbage collector has been developed in order to delete past context data from the knowledge base that is not relevant for the system (Section 3.3.4).

An additional requirement that has been considered in the design of the architecture is the extensibility of the platform, because it needs to be flexible enough in order to be extended at runtime. In such dynamic environments new context sources are required in order to identify new situations. This way, the data model of the platform and the defined rules can be extended according to the new context data requirements at runtime (Section 3.3.5).

Finally, the location context parameter has been specifically managed by the architecture. The entities that are involved in a certain situation can be on the move (e.g.

Person, Car), so the architecture of the platform integrates a GIS service in order to manage their location (Section 3.3.6).

The previously mentioned requirements have been taken into account in order to design the architecture of the implemented *Context Cloud* platform (Chapter 5). The improvements over the common architecture of so far prevalent context-aware development platforms are illustrated in the next figure.

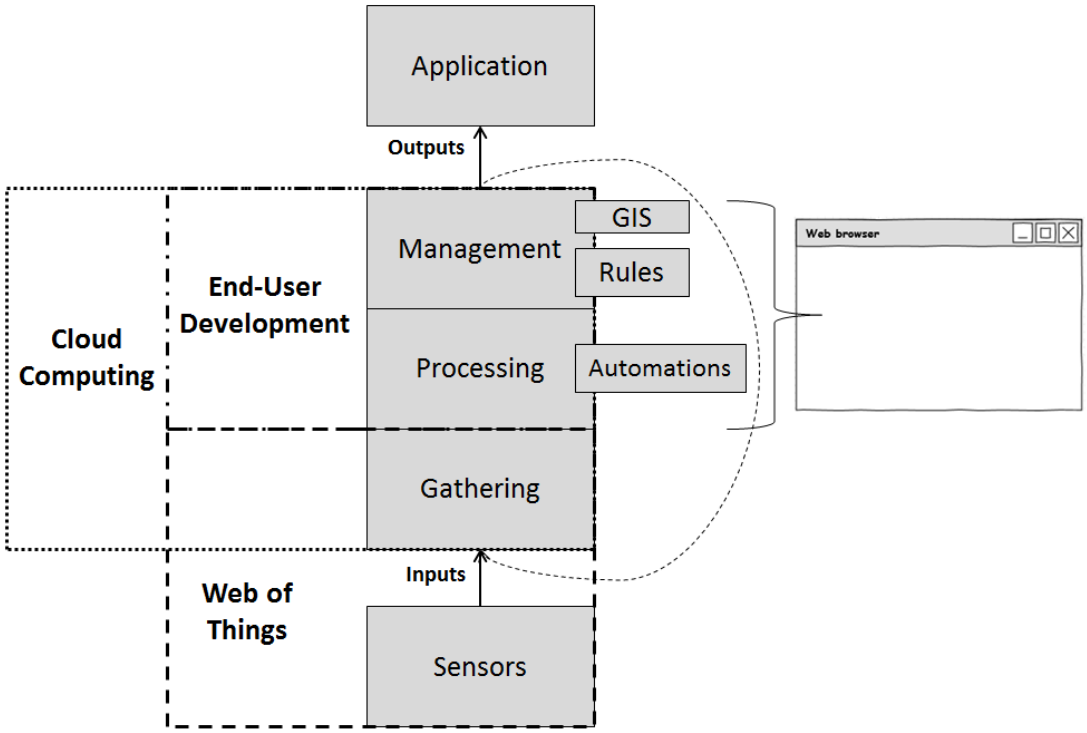


Fig. 86. Improvements over the common architecture of context-aware development toolkits

This new architecture for the design of context-aware software toolkits is a consequence of the achievement of the goal G2. This contribution is also part of the consecution of goal G4 and it has been based in the goal G3.

7.1.5. C5. A new methodology to guide the development process of context-aware systems in collaboration with domain experts

As mentioned before, there are no suitable development methodologies in order to be applied in the implementation of context-aware systems. This way, in order to guide the development process of context-aware systems and involve non-technical domain experts in collaboration with programmers, a methodology has been designed. This methodology is based on the definition of situations and it is based on the requirements

specified in Section 3.2. It is also inspired in agile software development practices, where software is incrementally developed based on well-defined tasks.

The aim of the designed methodology is to promote the collaboration among domain experts and programmers in the development of context-aware systems. This methodology considers the different situations of the entities that are relevant to adapt the behaviour of the system to be developed (Chapter 3). The methodology is divided into four different stages: analysis, configuration, development and validation.

In the *analysis stage*, domain experts and programmers have to identify all the situations that can be relevant for the system to be developed, specifying a name, a description and the desired behaviour of the system once the situation is detected. Also, each of the identified situations has to be parameterized with the entities that are involved in the situation, the location where the situation is detected and the interval of time when the situation can be detected. The needed inputs of context data in order to detect the situation have to be specified as well, providing the objective, the conditions and the restrictions for each data type. Finally, the needed outputs once a situation is detected have to be specified. These outputs are then used by programmers in order to adapt the system's behaviour according to the previous specifications. In order to support the analysis stage, an excel sheet template has been designed, where domain experts and programmers can discuss about the needed parameters of each of the identified situations.

Once the analysis stage is finished, the platform has to be configured with the specified parameters. In the *configuration* stage, the programmer has to identify and configure the context sources that can provide the defined inputs of data, and configure or implement the providers that are going to obtain these data from the identified sources. The next step is the creation of the areas where the situations can be detected, the context data model that will store context data, the mappings between the obtained data and the model, and the inference mechanisms in order to detect the needed situations. Domain experts with the collaboration of programmers should do these configurations, so that as mentioned before, the development platform provides configuration mechanisms in order to avoid the usage of programming languages.

Finally, the programmer has to *implement* the defined behaviours of the system to be developed, processing the outputs generated by the toolkit. Also, the system has to be tested and *validated* by domain experts and programmers.

As mentioned before, the aim of the methodology is to guide the development process of context-aware systems, promoting the collaboration of programmers and domain experts. This contribution establishes new guidelines to develop context-aware systems that improve the development process and the obtained final results. It has been achieved by the consecution of the goal G1 and it has been based on the goal G3. This contribution has been fundamental for the consecution of the goal G4 as well.

7.1.6. C6. A new platform to ease the development of context-aware systems in collaboration with domain experts

Section 2.8 analyses several toolkits and programming frameworks that have been proposed in order to simplify the development of context-aware applications. However, there are still some gaps in the reviewed frameworks.

These gaps are fulfilled with the requirements explained in Section 3.3. On the one hand, not all of them are designed to support users' mobility, and do not provide a GIS service in order to perform geospatial operations. This is crucial since users' location is the primary context parameter in context-aware scenarios. On the other hand, the reviewed frameworks offer high-level programming APIs for skilled programmers. This makes the involvement of non-technical stakeholders in the development life cycle, particularly domain experts, almost impossible. The participation of users that do not have programming skills but are experts in the application domain can speed up and improve the development process of context-aware services.

The aim of the platform has been to make the development of context-aware systems easier, even for people that do not have technical skills (Chapter 5). The platform provides a web front-end where all the features involved in the development of context-aware services can be easily configured without writing any line of code. For instance, the context data model, the context information gathering process and the rules to identify users' situations can be configured using the web interface. This way, the involvement of domain experts in the development process is possible.

Context Cloud can be considered as a black box that receives inputs from the identified context sources and produces outputs triggered by the defined context rules. These rules are used to identify situations according to the obtained context data and the defined data model.

The architecture of the system has been defined following the common features found in the reviewed frameworks and toolkits. Furthermore, its design has been based on the concept of End-user Programming paradigm, Cloud Computing infrastructures and the advances in the area of the Web of Things as mentioned in Section 7.1.4.

The platform differs from the reviewed toolkits in the following respects. It is designed to promote the collaboration among technical and non-technical users, guided by a development methodology. It provides a web environment where context data can be managed using a graphical interface without having to code anything, all of it from a standard web browser. It also provides geospatial functionalities to manage location context data and all the configurations can be extended at runtime.

Furthermore, *Context Cloud* presents more functionalities than the rest of the frameworks (Section 5.9). For instance, it is the only one that automatically manages context data life cycle. This management involves the automatic conversion from gathered raw context data to the context data model and automatic updates of current context data and storage of past context data. It also deletes context entities instances from the knowledge base when a registered context source is unregistered or it is no longer available. It is extensible, that is, the user can update the context model, the rules and the areas whenever it is needed in runtime. Also, a geographic information system is included in order to support user's mobility. Finally, *Context Cloud* can be configured using a web front-end, which makes it accessible from any Internet connected device.

The performance of the reasoning system of the platform has been validated as well (Section 6.2). This reasoning process could be a bottleneck in these kinds of systems. Based on the obtained results some advices have been stated in order to be taken into account when creating rules with the platform (Section 6.2.2). Like that, it is better to specify as many filters as possible in the rule conditions. Also, it is advisable to create as few entity instances as possible and create as few rules as possible. Join operations between different instances have to be avoided as far as possible as well. Finally, the server where the platform is deployed has to be configured with a high amount of memory because the reasoning engine requires high memory consumption. These execution restrictions can be easily matched by any cloud infrastructure where *Context Cloud* could be deployed. Attention has been paid to make the overall solution cloud compliant, to enable its flexible deployment to diverse Cloud infrastructures.

With the described platform, the goal G2 is completely achieved. This contribution is also essential for the completion of the goal G4.

7.1.7. C7. Involvement of domain experts in the development of context-aware systems in collaboration with programmers

The platform has been validated with 20 participants. They carried out the evaluation in pairs composed by a domain expert and a programmer. The evaluation was performed in a tourism scenario, so the non-technical users were experts in the tourism domain (Section 6.1.1) and the most relevant results have been already summarized (Section 6.2.2).

It is worth of mention that 95% of the participants found that learning the methodology was easy and that 100% of them stated that it eases the collaborative work. Also, 100% of the participants considered the methodology useful to work with the platform, and that it is useful to develop context-aware systems. In general terms, programmers learned the methodology faster and found it easier than domain experts.

These results evidence that the designed methodology can be applied to the development of context-aware systems. Furthermore, it can be used to promote the collaboration of domain experts with programmers in the development process. These findings contribute to the achievement of the goal G4.

Moreover, 95% of the participants indicated that learning how to use the platform is easy as well. Three out of four of the participants found it easy to get *Context Cloud* to do what they wanted to do. However, the other 25% disagreed on that. The reason is that the participants (mostly domain experts) were not familiarized at all with these kinds of toolkits. 90% of the domain experts stated that it would be easy for them to become skilful at using the platform.

The perceived utility of the platform was also highly supported by domain experts. 100% of the domain experts stated that using *Context Cloud* in their jobs would enable them to develop context-aware systems more quickly and that it would make it easier to develop context-aware systems. Also, all of the participants pointed out that they would recommend other users to use the platform and they would use it in future developments. In addition to this, 80% of them would be willing to pay for the system.

These findings evidence that the platform can be used in order to implement context-aware systems. It is clear that programmers become skilful users of the platform

easier than domain experts, but these also state that they could be skilful at using the platform easily as well. Furthermore, these results evidence that domain experts can be involved in the configuration of the platform, and thus, in the development process guided by the designed methodology. This statement is reinforced with the fact that 12 out of 40 situations were completely configured by domain-experts using the platform. These results contribute to the achievement of the goal G4.

The average time spent by each of the pair participants to solve the evaluation test was 90 minutes. It is relevant that they spent an average time of 37 minutes in order to solve the situation number one, while for the rest of the situations the average time was 17 minutes. This means that once they know how to configure the platform in order to identify the first situation, it is easier for them to configure it for the rest of the situations. This way, the learning curve is steep, that is, the participants learn in a very short period of time how to use the platform successfully.

All these findings have been achieved by the consecution of the goals G1 and G2, and they have led to the consecution of the goal G4 as well. This way, all the goals have been achieved and the obtained contributions evidence that **it is possible to involve domain experts in the development of context-aware systems with programmers, providing collaborative development methodologies adapted to people that do not have programming skills**.

7.2. Research and Scientific Implications

The new theory, the methodology and the devised development platform have several research and scientific implications.

The *context* and *situation* definitions have implications in the kind of information that needs to be taken into account when creating the model of context in order to develop a context-aware system. Also, they have implications in the design of new methodologies and toolkits that may assist developers in the implementation of context-aware systems.

Computing paradigms that complement Context-Aware Computing field, such as the Web of Things, End-User Development and Cloud Computing, also have several implications in the evolution of this research field. The advances may be applied in order to improve the state of the art of Context-Aware Computing and their supporting software infrastructures.

The development methodology, the platform, and the results obtained from the user evaluation have a fundamental implication in the evolution of Context-Aware Computing. This way, the involvement of domain-experts in the development process of context-aware systems is essential, which implies that the methodologies and tools used to implement such systems have to be adapted to people that do not have programming skills. This has a direct impact on both the design of the processes that have to be taken into account in the development of context-aware systems and the design of the toolkit architectures that support the development of such systems. It implies that programmers of systems that use context information must be aware of the importance of the involvement of domain-experts in early stages of the development process.

7.3. Future work

Despite the fact that this dissertation addresses a number of open issues in context-awareness, it has to necessarily set its scope and has therefore some limitations. In addition, evidence shown by experimentation has risen up new questions that deserve special research attention. Following are the open issues proposed to be addressed.

- *Research on context theory.* The notion of context can have different approaches based on the concrete domain where context itself is applied. This dissertation has stated a notion of context in order to be applied to the development of context-aware systems from a computer science perspective. It also has been used to model situations related to the tourism domain. There are as many facets for the notion of context as one can imagine. This suggests that the notion of context may be further studied and extended in order to be applied to different domains.

- *Distribution of the reasoning process.* Reasoning is usually performed in one single machine, be it the server or the client. The architecture presented in Chapter 4 considers that the reasoning process is carried out in a centralized way in a cloud infrastructure in order to take advantage of computing power of the hosted servers. But the question is: could the reasoning process be performed in a distributed way (different machines / different data centres) in order to improve the reasoning time?

- *Complementary reasoning algorithms*. The platform described in Chapter 4 considers rules in order to detect situations based on the context information gathered by the providers. As mentioned in Section 2.7, there are other algorithms that can be used in order to detect situations. Could any of these algorithms be used in order to complement the reasoning performed by the configured rules?

- *Uncertainty in context-aware reasoning*. When humans reason over certain facts, the decision taken is most of the time given a certain weight, i.e. a probabilistic value. There is a degree of uncertainty in the reasoning process that has not been considered in the implementation of the used reasoning engine for the purposes of this dissertation. However, being able to measure the probability with which a certain situation may happen is significant enough in context-aware computing and that future iterations of the development platform ought to look at.

- *Automatic rule creation*. Rule-based reasoning is based on the rules that have to be configured in advance by the users of the *Context Cloud* platform for a particular need. In this sense, is there any way to generate rules automatically? Can learning algorithms be applied to learn rules, and to derive new rules with base in what they have learnt?

- *Mobile battery life*. The location of users is one of the main context parameters to be considered in the development of context-aware systems. This information can be obtained from mobile devices, but the extraction of location coordinates from the GPS of mobile devices has a negative impact on the mobile phone's battery life because it has to be carried out periodically and it has to be sent to the platform. Could an algorithm be designed in order to improve the battery consumption of mobile devices?

- *Development of context-aware systems*. This dissertation presents a platform to ease the implementation of context-aware systems, where programmers and domain experts can collaborate in the development process. This way, domain experts have an active role in the development life cycle. Could the platform be improved in order to minimize the involvement of

programmers? Could domain-experts create context-aware systems without the need of programmers?

7.4. Final remarks

Context-aware computing is an extensive research field, in which knowledge and techniques from other complementary disciplines also have a relevant impact. Like that, there are still enormous opportunities to make significant contributions to the core body of knowledge of this discipline starting from the current status. Thus, a number of ambitious challenges have been set and faced at the beginning of this dissertation. A new theoretic approach to the notions of *context* and *situation* has been built. Then, based on the previous, new guidelines and requirements for the development of context-aware systems have been established. A new approach to the architecture of context-aware toolkits has been proposed, in order to foster the development of these kinds of systems by programmers and domain-experts. Based on this architecture, a platform to develop context-aware systems has been implemented as well. Also, a new methodology to guide the development process of context-aware systems has been proposed, where programmers and domain-experts can collaborate in order to improve the development life cycle, and thus, the final product.

The evolution of mobile devices and their proliferation in society, the advancement of connectivity technologies and embedded sensors, and the trend towards creating hybrid spaces (symbiosis between nature and technology) will trigger a radical change in the way persons involved in mobility interact with their environment and consume information and services. In such scenario, the usage of context information is essential in order to personalize the information consumption and to improve the user experience.

The results of this dissertation are expected to have a significant impact on context-aware computing in general and on the development of context-aware systems in particular, that boost the development of smarter systems and the relationship of humans with their ever more digital environment by the use of such new systems. Therefore, context-based systems will also provide with the opportunity to better understand human behaviour in the digital society of the future.

With the theory set forward in the framework of this dissertation and its resulting prototype, I hope to have contributed to the status of core knowledge of context-aware

computing and to set the basis for consistent future research and scientific activities based on the designed *Situation-Driven Development* methodology and the *Context Cloud* platform.

Referencias

Bibliografía

Abowd, G. D., Atkeson, C. G., Hong, J., Long, S., Kooper, R., & Pinkerton, M. (1997). Cyberguide: A Mobile Context-Aware Tour Guide. *Wireless Networks*, *3*(5), 421-433.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, *26*(11), 832-843. doi:10.1145/182.358434

Bacon, J., Bates, J., & Halls, D. (1997). Location-oriented multimedia. *Personal Communications*, *4*(5), 48-57.

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, *2*(4), 263-277. doi:10.1504/IJAHUC.2007.014070

Bardram, J. E. (2005). The Java Context Awareness Framework (JCAF) – A Service Infrastructure and Programming Framework for Context-Aware Applications. In H. Gellersen, R. Want, & A. Schmidt (Eds.), *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)* (Vol. 3468, pp. 98-115). Munich, Germany: Springer Verlag.

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change*. Pearson Education Inc.

Beer, T., Rasinger, J., Höpken, W., & Fuchs, M. (2007). Exploiting E-C-A Rules for Defining and Processing Context-Aware Push Messages. In Springer (Ed.), *Proceedings of the International RuleML Symposium on Rule Interchange and Applications* (Vol. 4824, pp. 199-206). Orlando.

Bellotti, V., & Edwards, K. (2001). Intelligibility and Accountability: Human Considerations in Context-Aware Systems. *Human-Computer Interaction*, *16*(2), 193-212. doi:10.1207/S15327051HCI16234_05

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., & Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, *6*(2), 161-180. doi:10.1016/j.pmcj.2009.06.002

Bikakis, A., Patkos, T., Antoniou, G., & Plexousakis, D. (2008). A Survey of Semantics-based Approaches for Context Reasoning in Ambient Intelligence. *Proceedings of Constructing Ambient Intelligence - AmI-07 Workshops* (pp. 14-23). Springer.

Bishop, T. A., & Karne, R. K. (2003). A Survey of Middleware. *Computers and Their Applications* (pp. 254–258).

Bizer, C. (2009). The Emerging Web of Linked Data. *IEEE Intelligent Systems*, *24*(5), 87-92.

Brézillon, P. (2000). *Modeling and Using Context : Past , Present and Future. Knowledge Acquisition*.

Brown, P. J. (1996). The stick-e Document: A Framework for Creating Context-Aware Applications. In J. W. Sons (Ed.), *Proceedings of the International Conference on Electronic Documents* (Vol. 8, pp. 259-272). New York.

Bucur, O., Beaune, P., & Boissier, O. (2005). Representing Context in an Agent Architecture for Context-Based Decision Making. *Proceedings of the Workshop on Context Representation and Reasoning (CRR '05)*. Paris.

Buriano, L. (2006). Exploiting Social Context Information in Context-Aware Mobile Tourism Guides. *Proceedings of Mobile Guide*.

Cassou, D., Bruneau, J., & Consel, C. (2010). A tool suite to prototype pervasive computing applications. *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)* (pp. 820-822). IEEE doi:10.1109/PERCOMW.2010.5470550

Charles Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", *Artificial Intelligence*, 19, pp 17–37, 1982

Chen, G., & Kotz, D. (2000). *A Survey of Context-Aware Mobile Computing Research*. *Time*. (Technical Report Number TR2000-381). Dartmouth College, Department of Computer Science, UK.

Chen, H. L. (2004). *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. *Interfaces*. University of Maryland, Baltimore Counry.

Chen, H., County, B., Finin, T., & Joshi, A. (2003). Semantic Web in a Pervasive Context-Aware Architecture. *Proceedings of the Artificial Intelligence in Mobile System (AIMS 2003)* (pp. 33-40).

Conlan, O., Power, R., Higel, S., Sullivan, D. O., & Barrett, K. (2003). Next Generation Context Aware Adaptive Services. *Proceedings of the 1st international symposium on Information and communication technologies*.

Cook, D. J., Augusto, J. C., & Jakkula, V. R. (2009). Ambient Intelligence : Technologies, Applications, and Opportunities. *Pervasive and Mobile Computing*, *5*(4), 277-298.

Costabile, M. F., Fogli, D., Letondal, C., Mussio, P., & Piccinno, A. (2003). Domain-Expert Users and their Needs of Software Development. *UAHCI Conference* (pp. 232–236).

Coutaz, J., & Rey, G. (2002). Foundations for a Theory of Contextors. *Proceedings of the 4th International Conference on Computer-Aided Design of User Interfaces (CADUI '02)* (pp. 283-302). France: Kluwer Academics Publishing.

Davis, F. D. (1989). Perceived Usefulness , Perceived Ease of Use , and User Acceptance of lnformation Technology. *MIS Quarterly*, *13*(3), 318-340.

Dey, A. K. (2000). *Providing Architectural Support for Building Context-Aware Applications*. *Ph.D. Thesis*. Georgia Institute of Technology.

Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing*, *5*(1), 4-7. doi:10.1007/s007790170019

Dey, A., Abowd, G., & Salber, D. (2001). A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. *Human-Computer Interaction*, *16*(2), 97-166. doi:10.1207/S15327051HCI16234_02

Fahy, P., & Clarke, S. (2004). CASS - Middleware for Mobile Context-Aware Applications. *Workshop on Context Awareness, MobiSys*.

Gajos, K., Fox, H., & Shrobe, H. (2002). End User Empowerment in Human Centered Pervasive Computing. *Proceedings of Pervasive* (pp. 134-140).

Green, D., & DiCaterino, A. (1998). *A Survey of System Development Process Models*. Center for Technology in Government University. Albany.

Gross, T., & Specht, M. (2001). Awareness in Context-Aware Information Systems. *Proceedings of the Mensch und Computer* (pp. 173-181). Stuttgart.

Gruber, T. R., & Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications. *Knowledge Creation Diffusion Utilization*, *5*(2), 199-220.

Gu, T., Pung, H. K., & Zhang, D. Q. (2004). A middleware for building context-aware mobile services. *Proceedings of the 59th Vehicular Technology Conference (VTC)* (Vol. 5, pp. 2656-2660). Ieee. doi:10.1109/VETECS.2004.1391402

Guan, D., Yuan, W., Lee, S., & Lee, Y.-K. (2007). Context Selection and Reasoning in Ubiquitous Computing. *Proceedings of the 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)* (pp. 184-187). Ieee. doi:10.1109/IPC.2007.102

Guesgen, H. W., y Marsland, S. (2010). Spatio-Temporal Reasoning and Context Awareness. In H. Nakashima, H. Aghajan, & J. C. Augusto (Eds.), *Handbook of Ambient Intelligence and Smart Environments* (pp. 609-634). Boston, MA: Springer US. doi:10.1007/978-0-387-93808-0

Guinard, D., Trifa, V., Mattern, F., & Wilde, E. (2011). From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices. In D. Uckelmann, M. Harrison, & F. Michahelles (Eds.), *Architecting the Internet of Things* (pp. 97-129). Springer.

Guo, B., Zhang, D., & Imai, M. (2010). Toward a cooperative programming framework for context-aware applications. *Personal and Ubiquitous Computing*, *15*(3), 221-233. doi:10.1007/s00779-010-0329-1

Gwizdka, J. (2000). What 's in the Context? *Computer Human Interaction (CHI 2000)*.

Harel, D. and Pnueli, A. (1985). On the development of reactive systems. *Logics and models of concurrent systems* (pp. 477-498). New York: Springer-Verlag.

Hayes, B. (2008). Cloud computing. *Communications of the ACM*, *51*(7), 9. doi:10.1145/1364782.1364786

Henricksen, K. (2003). *A Framework for Context-Aware Pervasive Computing Applications*. University of Queensland.

Henricksen, K., & Indulska, J. (2006). Developing Context-Aware Pervasive Computing Applications: Models and Approach. *Pervasive and Mobile Computing*, *2*(1), 37-64.

Hess, C. K., & Campbell, R. H. (2003). An application of a context-aware file system. *Personal and Ubiquitous Computing*, *7*(6), 339-352. doi:10.1007/s00779-003-0250-y

Hirschfeld, R., & Costanza, P. (2008). Context-oriented Programming. *Journal of Object Technology*, *7*(3), 125-151.

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., & Altmann, J. (2002). Context-awareness on mobile devices - the hydrogen approach. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences* (pp. 292-302).

Hong, J.-yi, Suh, E.-ho, & Kim, S.-J. (2009). Context-aware systems: A literature review and classification. *Expert Systems with Applications*, *36*(4), 8509-8522. Elsevier Ltd. doi:10.1016/j.eswa.2008.10.071

Hull, R., Neaves, P., & Bedford-roberts, J. (1997). Towards Situated Computing. *1st IEEE International Symposium on Wearable Computers (ISWC '97)*.

Indulska, J., & Sutton, P. (2003). Location management in pervasive systems. *Proceedings of the Australasian information security workshop conference on ACSW frontiers*. Darlinghurst: Australian Computer Society.

Ipina, D., & Katsiri, E. 2001. An ECA Rule-Matching Service for Simpler Development of Reactive Applications. Published as a supplement to the Proc. of Middleware 2001 at IEEE Distributed Systems Online, Vol. 2, No. 7.

Kasteren, T. V., Noulas, A., Englebienne, G., & Kr, B. (2008). Accurate Activity Recognition in a Home Setting. *Proceedings of the 10th International Conference on Ubiquitous Computing*. Seoul: ACM.

Korpipää, P., Mantyjarvi, J., Kela, J., Keranen, H., & Malm, E.-j. (2003). Managing context information in mobile devices. *Pervasive Computing*, *2*(3), 42-51. doi:http://dx.doi.org/10.1109/MPRV.2003.1228526

Lamsfus, C. (2010). *CONCERT : A New Framework for Contextual Computing in Tourism to Support Human Mobility*. Tesis Doctoral, Facultad de Ingeniería, Universidad de Deusto.

Lamsfus, C., Alzua, A., Martin, D., & Smithers, T. (2011). An Evaluation of a Contextual Computing Approach to Visitor Information Systems. In R. Law, M. Fuchs, & F. Ricci (Eds.), *Information and Communication Technologies in Tourism 2011: Proceedings of the International Conference in Innsbruck* (pp. 179-190). Springer Vienna.

Lee, D., & Meier, R. (2007). Primary-Context Model and Ontology: A Combined Approach for Pervasive Transportation Services. *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07)*, 419-424. Ieee. doi:10.1109/PERCOMW.2007.95

Lieberman, H., & Selker, T. (2000). Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal*, *39*(3), 617-632. doi:10.1147/sj.393.0617

Lieberman, H., Paternó, F., & Klann, M. (2006). End-User Development: an Emerging Paradigm. *End User Development*, *9*, 1-8.

Marzano, S., & Aarts, E. (2003). *The New Everyday View on Ambient Intelligence* (p. 365). Uitgeverij 010.

McCarthy, J., & Hayes, P. J. (1968). *Some philosophical problems from the standpoint of artificial intelligence*. Stanford, California.

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proc. AFIPS Fall Joint Computer Conference* (Vol. 33, pp. 267-277).

Nielsen, J., & Rolf, M. (1990). Heuristic evaluation of User Interfaces. *Proceedings of the SIGCHI conference on human factors in computing systems: Empowering people* (Vol. 17, pp. 249-256). Seattle. doi:10.1089/tmj.2010.0114

Oh, Y., Yoon, H., & Woo, W. (2006). Simulating Context-Aware Systems based on Personal Devices. *Proceedings of the International Symposium on Ubiquitous VR* (pp. 49-52).

Oppermann, R. (2005). From User-adaptive to Context-adaptive Information Systems. *(Von benutzeradaptiven zu kontextadaptiven Informationssystemen) i-com*, *4*(3), 4-14.

Patterson, D. J., Liao, L., Fox, D., & Kautz, H. (2003). Inferring High-Level Behavior from Low-Level Sensors. *Proceedings of the 5th International Conference on Ubiquitous Computing* (pp. 73-89). Berlin: Springer.

Rahlff, O.-W., Kenneth Rolfsen, R., & Herstad, J. (2001). Using Personal Traces in Context Space: Towards Context Trace Technology. *Personal and Ubiquitous Computing*, *5*(1), 50-53. doi:10.1007/s007790170030

Ranganathan, a., Al-Muhtadi, J., & Campbell, R. H. (2004). Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, *3*(2), 62-70. doi:10.1109/MPRV.2004.1316821

Riboni, D., & Bettini, C. (2010). COSAR: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, *15*(3), 271-289. doi:10.1007/s00779-010-0331-7

Rossi, G., & Sánchez-Figueroa, F. (2010). Rich Internet Applications. *IEEE Internet Computing*, *14*(3), 9-12.

Ryan, N. (1999). ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server.

S. Steiniger, M. Neun, and A. Edwardes, "Foundations of Location Based Services", Lecture Notes on LBS, 2006, Department of Geography, University of Zürich.

Sadeh, N. M., Gandon, F. L., & Kwon, O. B. (2005). *Ambient Intelligence: The MyCampus Experience*. School of Computer Science, Carnegie Mellon University. Technical Report CMU-ISRI-05-123

Scharff, C., & Verma, R. (2010). Scrum to support mobile application development projects in a just-in-time learning context. *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering - CHASE'10*, 25–31. doi:10.1145/1833310.1833315

Schilit, B. N., & Theimer, M. (1994). Disseminating Active Map Information to Mobile Hosts. *IEEE Network*, *8*(5), 22-32.

Schilit, B., Adams, N., & Want, R. (1994). Context-aware computing applications. *Proceedings of the Workshop on Mobile Computing Systems and Applications* (pp. 85-90). Santa Cruz: IEEE Comput. Soc. Press. doi:10.1109/MCSA.1994.512740

Schmidt, A., & van Laerhoven, K. (2001). How to build smart appliances? *IEEE Personal Communications*, *8*(4), 66-71.

Schmidt, A., Asante, K., Takaluoma, A., Tuomela, U., Laerhoven, K. V., & Velde, W. V. D. (1999). Advanced Interaction in Context. In S. Verlag (Ed.), *Proceedings of First International Symposium on Handheld and Ubiquitous Computing* (pp. 89-101).

Schmidt, A., Beigl, M., & Gellersen, H. W. (1999). There is more to context than location. *Computers and Graphics*, *23*, 893-901. doi:10.1016/S0097-8493(99)00120-X

Sohn, T. Y., & Dey, A. K. (2003). iCAP: An Informal Tool for Interactive Prototyping of Context-Aware Applications. *Proceedings of Extended abstracts on Human factors in computing systems* (p. 974--975). New York: ACM Press.

Steiniger, S., Neun, M., & Edwardes, A. (2006). Foundations of Location Based Services. *Lecture Notes on LBS*. Department of Geography, University of Zürich.

Storf, H., Becker, M., & Riedl, M. (2099). Rule-based Activity Recognition Framework: Challenges, Technique and Learning. *Proceedings of the 3rd International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)* (pp. 1-7). London.

Strang, T., & Linnhoff-popien, C. (2004). A Context Modeling Survey. *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*. Nottingham.

Sumi, Y., Etani, T., Fels, S., Simonet, N., Kobayashi, K., & Mase, K. (1998). C-map: Building a context-aware mobile assistant for exhibition tours. *Community Computing and Support Systems, Social Interaction in Networked Communities* (pp. 137-154). London: Springer Verlag.

Tang, Z., Hile, H., Bajracharya, S., & Jurdak, R. (2005). PetTracker – Pet Tracking System Using Motes. *Proceedings of the Seventh International Conference on Ubiquitous Computing (Ubicomp)*.

Tobler, W. R. (1970). A computer model simulation of urban growth in the Detroit region. *Economic Geography*, *46*(2), 234-240.

Trigg, R., Moran, T., & Halasz, F. (1987). Adaptability and Tailorability in NoteCards. In B. Bullinger, H.-J. and Shackel (Ed.), *Proceedings of INTERACT '87* (pp. 723–728). London.

Wang, X. H., Gu, T., Zhang, D. Q., & Pung, H. K. (2004). Ontology Based Context Modeling and Reasoning using OWL. *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04)* (pp. 18-22). IEEE Computer Society.

Wang, X., Dong, J. S., Chin, C. Y., Hettiarachchi, S. R., & Zhang, D. (2004). Semantic Space: An Infrastructure for Smart Spaces. *IEEE Pervasive Computing*, *03*(03), 32-39. doi:10.1109/MPRV.2004.1321026

Want, R., Hopper, A., Falcao, V., & Gibbons, J. (1992). The active badge location system. *Transactions on Information Systems*, *10*(1), 91-102. doi:http://doi.acm.org/10.1145/128756.128759

Weiser, M. (1991). The computer for the 21st century. *Scientific American*, *265*(3), 94-104.

Whitehead, J. (2007). Collaboration in Software Engineering: A Roadmap. *Proceedings of the 7th Future of Software Engineering (FOSE)*. (pp. 214-225). IEEE Computer Society.

Yau, S. S., & Huang, D. (2006). Mobile Middleware for Situation-Aware Service Discovery and Coordination. In P. B. and A. Corradi (Ed.), *Handbook of Mobile Middleware*.

Ye, J., Dobson, S., & McKeever, S. (2011). Situation identification techniques in pervasive computing. *Pervasive and Mobile Computing*. Elsevier B.V. doi:10.1016/j.pmcj.2011.01.004

Zhang, T., & Brügge, B. (2004). Empowering the User to Build Smart Home Applications. *Proceedings of the Second International Conference On Smart homes and health Telematics*. Singapore.

Zhang, P., & Li, N. (2004). An assessment of human-computer interaction research in management information systems: topics and methods. *Computer in Human Behavior (CHB), 20*(2), 125-147.

Zhou, F., Jiao, J. R., & Chen, S. (2011). A Case-Driven Ambient Intelligence System for Elderly in-Home Assistance Applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, *41*(2), 179-189.

Informes

Oracle Communications. (2011). *Opportunity Calling: The Future of Mobile Communications – Take Two. Middle East*. Disponible en:

http://www.oracle.com/us/industries/communications/oracle-communications-future-mobile-521589.pdf Último acceso 19-03-2012.

Sitios web

Cosm

https://cosm.com/ Último acceso 26-07-2012.

Drools Expert

http://www.jboss.org/drools/drools-expert.html Último acceso 11-04-2012.

Drools Fusion

http://docs.jboss.org/drools/release/5.4.0.CR1/drools-fusion-docs/html/ch02.html#d0e547 Último acceso 13-04-2012

Gartner

http://www.gartner.com/technology/research/top-10-technology-trends/ Último acceso 26-03-2012.

Google developers

https://developers.google.com/maps/?hl=es Último acceso 29-03-2012.

Java-source

http://java-source.net/open-source/rule-engines Último acceso 22-08-2012

JCP

http://www.jcp.org/en/jsr/detail?id=94 Último acceso 22-08-2012.

Nimbits

http://www.nimbits.com Último acceso 26-07-2012.

Oracle

http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Number.html Último acceso 12-04-1982.

http://www.oracle.com/technetwork/java/javame/index.html Último acceso 26-07-2012

Overlay Media

http://www.overlaymedia.com/ Último acceso 16-04-2012.

Pervasive Media Studio

http://www.pmstudio.co.uk/ Último acceso 16-04-2012.

PostgreSQL

http://www.postgresql.org.es/ Último acceso 29-03-2012.

Programmable Web

http://www.programmableweb.com/ Último acceso 26-07-2012.

Siafu simulator

http://siafusimulator.sourceforge.net/ Último acceso 15-05-2012.

Technodec

http://techondec.wordpress.com/2011/03/14/rete-algorithm-demystified-part-2/ Último acceso 08-05-2012.

TICBeat

http://www.ticbeat.com/wp-content/uploads/2011/04/mobile_infographic.jpg Último acceso 19-03-2012.

Yahoo weather

http://developer.yahoo.com/weather/ Último acceso 29-03-2012.

Anexos

En este capítulo se muestran los anexos referentes a la (i) lista de publicaciones realizadas durante el transcurso de la presente investigación, así como aspectos relacionados con la implementación de la plataforma descrita en el Capítulo 4, como son (ii) la sintaxis de reglas utilizada por el motor de razonamiento, (iii) la descripción de las funcionalidades del simulador *Siafu* utilizado en las pruebas de validación, así como (iv) el cuestionario de evaluación presentado a los participantes.

Anexo I. Lista de publicaciones

Durante el período de investigación, se validó el trabajo elaborado mediante la presentación de diferentes artículos en conferencias enmarcadas en el campo de los sistemas sensibles al contexto. La siguiente lista muestra las publicaciones más relevantes en las que se ha participado.

- **Martín, D.**, López-de-Ipiña, D., Lamsfus, C., & Alzua, A. (2012). Situation-Driven Development: a Methodology for the Development of Context-Aware Systems. *Proceedings of the 6th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI)* (pp. 241-248). Vitoria-Gasteiz (Spain).

- **Martín, D.**, Lamsfus, C., Alzua, A., & López-de-Ipiña, D. (2012). A Web Platform and a Methodology to promote a Collaborative Development of Context-Aware Systems. *Proceedings of the 8th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (pp. 126-133). Barcelona (Spain).

- **Martín, D.**, Lamsfus, C., Alzua, A., & López-de-Ipiña, D. (2012). Foundations for a Platform to Develop Context-Aware Systems by Domain Experts. *Proceedings of the 11th IEEE International Conference on Ubiquitous Computing and Communications (IUCC)* (pp. 2029-2034). Liverpool. doi: 10.1109/TrustCom.2012.164.

- **Martín, D.**, López-de-Ipiña, D., Lamsfus, C., & Alzua, A. (2012). Involving tourism domain experts in the development of context-aware mobile services. *e-Review of Tourism Research, eRTR.*

- Lamsfus, C., **Martín, D.**, Alzua, A., López-de-Ipiña, D., & Torres, E. (2012). Context-Based Tourism Information Filtering with a Semantic Rule Engine. *Sensors, 12*(5), 5273-5289. doi: 10.3390/s120505273.

- **Martín, D.**, Lamsfus, C., & Alzua, A. (2011). A Contextual Geofencing Mobile Tourism Service. *Information and Communication Technologies in Tourism 2011* (pp. 191-202). Springer Verlag. Innsbruck (Austria). doi: 10.1007/978-3-7091-0503-0_16.

- **Martín, D.**, Lamsfus, C., & Alzua, A. (2011). Mobile context data management framework. *Proceedings of the 5th IEEE/FTRA International Conference on Multimedia and Ubiquitous Engineering (MUE)* (pp. 73-78). Greece. doi: 10.1109/MUE.2011.24

- Lamsfus, C., Alzua-Sorzabal, A., **Martín, D.**, & Torres-Manzanera, E. (2011). Semantic-based Context Modelling in Tourism. *Information Technology and Tourism*, *13*(4), 309-325. doi: 10.3727/109830512X13364362859894

- Lamsfus, C., **Martín, D.**, Alzua, A., & López-de-Ipiña, D. (2011). Towards Context-Aware Push/Filter Information Dissemination. *Proceedings of the 5th International Symposium on Ubiquitous Computing and Ambient Intelligence (UCAmI)*. Riviera Maya (Mexico).

- Lamsfus, C., Alzua, A., **Martín, D.**, & Tim Smithers. (2011). An Evaluation of Contextual Computing Approach to Visitor Information Systems. *Information and Communication Technologies in Tourism 2011* (pp. 179-190). Springer Verlag. Innsbruck, (Austria). doi: 10.1007/978-3-7091-0503-0_15.

- Lamsfus, C., Alzua, A., **Martín, D.**, & Zigor Salvador (2010). Semantic-based Contextual Computing support for Human Mobility. *Information and Communication Technologies in Tourism 2010* (pp. 603-615). Springer Verlag. Lugano (Switzerland). doi: 10.1007/978-3-211-99407-8_50.

- **Martín, D.**, Lamsfus, C., & Alzua, A. (2010). Automatic Context Data Life-Cycle Management Framework. *Proceedings of the 5th IEEE International Conference on Pervasive Computing and Applications (ICPCA)* (pp. 330-335). Maribor (Slovenija). doi: 10.1109/ICPCA.2010.5704122.

- Lamsfus, C., Alzua, A., **Martín, D.**, & López-de-Ipiña, D. (2010). Digital Broadcasting for Context-Aware Services in Tourism. *Proceedings of the 2nd Region IEEE Conference on the History of Telecommunications (HISTELCON)*. Spain.

- Cadenas, A., Ruiz, C., Larizgoitia, I., García-Castro, R., Lamsfus, C., Vázquez, I., González, M., **Martín, D.**, & Poveda, M. (2009). Context management in mobile environments: a semantic approach. *Proceedings of the*

1st Workshop on Context, information and ontologies (CIAO). Heraklion (Greece). doi: 10.1145/1552262.1552264.

- Lamsfus, C., Alzua, A., **Martín, D.**, Salvador, Z., & Usandizaga, A. (2009). Human-Centric Ontology-based Context Modelling in Tourism. *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD)* (pp. 424-434). Madeira (Portugal).

- Lamsfus, C., Alzua, A., **Martín, D.**, Salvador, Z., & Usandizaga, A. (2009). Contextual Computing based Services in Tourism. *Proceedings of the 4th Mediterranean Conference in Information Systems (MCIS)*. Athens (Greece).

El siguiente listado muestra otras publicaciones que no están directamente relacionadas con el trabajo realizado, pero que han tenido relevancia para la formación y trayectoria investigadora.

- Herrero, G., Campo, A., Gil, M., García, A., **Martín, D.**, Zugasti, I., Bilbao, S., Perez, A., Koshutanski, H., Maña, A., & Pérez, I. (2012). ConTur: an intelligent content management system for the tourism sector. *Information and Communication Technologies in Tourism 2012*. Helsingborg (Sweden).

- Lamsfus, C., **Martín, D.**, Alzua, A., López-de-Ipiña, D., & Torres-Manzanera, E. (2012). Intelligent Tourism Information Consumption: A Push Semantic Rule-based System. *Advances in Knowledge-Based and Intelligent Information and Engineering Systems* (Vol. 243, pp. 823-832). Donostia-San Sebastián (Spain). doi: 10.3233/978-1-61499-105-2-823.

- Buján, D., **Martín, D.**, Torices, O., & Uriarte, A. (2012). New Approaches in Context Modelling for Tourism Applications. *Proceedings of the 6th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI)* (pp. 379-386). Vitoria-Gasteiz (Spain).

Anexo II. Sintaxis de reglas

Las reglas pueden ser creadas mediante los controles proporcionados por la plataforma, además de poder ser codificadas de manera manual. El motor de reglas utilizado como base dispone de una sintaxis bien definida para la creación de reglas que resulta entendible a simple vista. A continuación se muestra la sintaxis básica para la creación de reglas.

Comentarios

El cuerpo de las reglas puede tener comentarios de una línea, como se muestra en el ejemplo de la línea 1, o de múltiples líneas, tal y como se muestra en las líneas 2, 3 y 4. Estos comentarios pueden ser utilizados para documentar las reglas creadas así como para especificar detalles específicos sobre las mismas.

```
1.  //Comentario de una línea
2.  /* Comentario realizado
3.  en
4.  múltiples líneas  */
```

Condiciones

A la hora de establecer las condiciones de la regla, se utilizan las entidades de contexto que conforman el modelo de datos creado y más concretamente sus propiedades. En el siguiente recuadro se muestra una condición sobre las instancias de entidades de tipo "Persona".

```
5.  Persona(nombre == "David", fechaNacimiento > 1982)
```

Como ejemplo, en la línea 5 se han especificado dos condiciones aplicables a las instancias de entidad "Persona" que se encuentren en el sistema. De esta manera, la propiedad del nombre tiene que ser igual a "David" y la fecha de nacimiento (propiedad "fechaNacimiento") tiene que ser posterior al año 1982 para que la condición se cumpla y la regla se ejecute.

Además, pueden establecerse tantas condiciones como sean requeridas, tal y como se muestra en las líneas 6, 7 y 8.

```
6.  Persona(nombre == "David", fechaNacimiento > 1982)
7.  Producto(precio == (antiguoPrecio * 1.10))
8.  Ciudad(temperatura > 25)
```

Variables

Una variable sirve para almacenar instancias o valores de las propiedades de las entidades participantes en una regla que se quieran utilizar en otros puntos de la regla. De esta manera, se pueden asignar tanto instancias de entidades como propiedades de las mismas a variables para su posterior uso en la regla. Las variables pueden nombrarse de cualquier manera, pero por convención se les antepone el prefijo "$". Así, en la línea 9 se asigna a una variable la edad de la entidad "Persona" cuyo nombre es "David" y además se asigna la propia instancia de "Persona" que cumple ese filtro a la variable "$persona". Estas variables pueden ser utilizadas en posteriores condiciones, como es el caso de la línea 10, donde se realiza una comparación accediendo a la propiedad "nombre" de la entidad "Persona".

```
9.  $persona : Persona (nombre ==”David”, $edad : edad)
10. $coche : Coche(nombre == $persona.nombre)
```

Operadores lógicos

Se pueden utilizar operadores lógicos a la hora de establecer condiciones (*and*, &&, *or*, ||, *not*) tal y como se muestra en el ejemplo. Cabe destacar, que si los operadores están dentro de una condición, como el caso de la línea 11, se utilizarán || y &&. Sin embargo, si los operadores están entre condiciones, se utilizarán *or* y *and* como muestra en las líneas 12 y 13.

En la línea 14 se muestra un ejemplo de utilización del operador *not*, donde se establece que no deben existir coches matriculados con anterioridad a 1998 para que la regla se ejecute.

```
11. Ciudad(temperatura > 25 && <35 || humedad == 75)
12. $persona : (Persona (nombre == “David”, edad < 18)
13. or Persona(nombre==”Juan”, edad < 28))
14. not Coche(añoMatriculacion < 1998)
```

Cláusula "eval"

La cláusula "eval" sirve para evaluar ciertas expresiones en la condición de una regla. El resultado de la evaluación tiene que ser "cierto" o "falso". En el ejemplo de la línea 15 se evaluará la condición dispuesta entre paréntesis donde la suma aritmética tiene que ser igual a 5. En este ejemplo dicha condición se cumple.

```
15. eval(2+3==5)
```

Cláusula "exists"

Esta cláusula sirve para especificar en la condición, la existencia de una serie de instancias de entidades que coincidan con un patrón determinado.

```
16. exists ( Bus(color == "rojo", numero == 42) )
17. /* Su significado es "si existe algún bus que cumple esas condiciones"… */
```

Cláusula "this"

Con "this" se hace referencia a la entidad que se trata en la propia condición. En el ejemplo se muestra una condición mediante las líneas 18 y 19, donde se almacenan en dos variables, dos personas que no son la misma.

```
18. $persona1 : Persona()
19. $persona2 : Persona(this != $persona1)
```

Cláusula "forall"

Con esta cláusula se especifica la condición para que todas las instancias de un tipo de entidad concreta cumplan con una serie de filtros. Por ejemplo, en la línea 20 se especifica que todas las "Personas" tienen que tener una edad mayor que 18 años.

```
20. forall( Persona( edad > 18 ) )
```

Un ejemplo más complejo es el que se muestra a continuación, donde se establece la condición de que todos los propietarios de tipo "Persona" de instancias de entidad "Coche" deben tener una edad mayor o igual a 18 años.

```
21. forall(
22.     $p : Persona(edad >= 18)
23.     Coche( propietario == $p )
24. )
```

Colecciones

Las propiedades que se asignan a las entidades pueden ser de tipo *List*, es decir, que pueden ser colecciones de entidades. Estas colecciones, admiten una serie de instrucciones como las que se presentan a continuación.

- *(not) contains*: determina que una colección contenga (o no) una instancia de una entidad determinada. En el ejemplo de la línea 26, se tiene una entidad "Recinto" que contiene una propiedad "listaPersonas" en la que se almacenan las personas que están en el interior del recinto. La condición configurada sirve para detectar a una persona determinada en el listado de personas.

```
25. $persona : Persona(nombre = "David")
26. Recinto (listaPersonas contains $persona)
```

- *(not) memberOf*: comprueba que una entidad sea miembro (o no) de una colección determinada. Así, en la línea 28 se establece la condición de que la entidad "Persona" debe estar en la lista de personas de la entidad "Recinto".

```
27. $recinto : Recinto()
28. Persona(this memberOf $recinto.listaPersonas)
```

- Acceso a elementos: se puede acceder de manera directa a los elementos de una colección mediante la sintaxis que se muestra en la línea 29, donde se accede al primer elemento de la colección "hijos" perteneciente a la entidad "Persona".

```
29. Persona(hijos[0].edad == 18 )
```

- *from*: establece condiciones sobre una colección determinada de entidades. En el ejemplo se tiene una colección de entidades de tipo "Producto" las cuales se filtran aquellas cuyo precio es mayor que 100.

```
30. $pedido : Pedido()
31. $producto  : Producto( precio > 100 ) from $pedido.productos
```

- *collect*: establece condiciones sobre una colección determinada de entidades previamente filtradas y recolectadas en una lista de tipo *ArrayList* en base a una serie de condiciones. En el siguiente ejemplo se puede observar como en la línea 34 se recolectan entidades de tipo "Alarma" que cumplen una serie de condiciones concretas y que se van a almacenar en la variable "$alarmas" en caso de que se encuentren tres o más "Alarmas" con dichas condiciones.

```
32. $sistema : Sistema()
33. $alarmas : ArrayList( size >= 3 )
34.         from collect( Alarma( sistema == $sistema, estado == "pendiente" ) )
```

Este tipo de condiciones se pueden configurar también con el diálogo para la generación de reglas, con la función *collect* y especificando un nombre de variable para recolectar las entidades, tal y como muestra la figura.

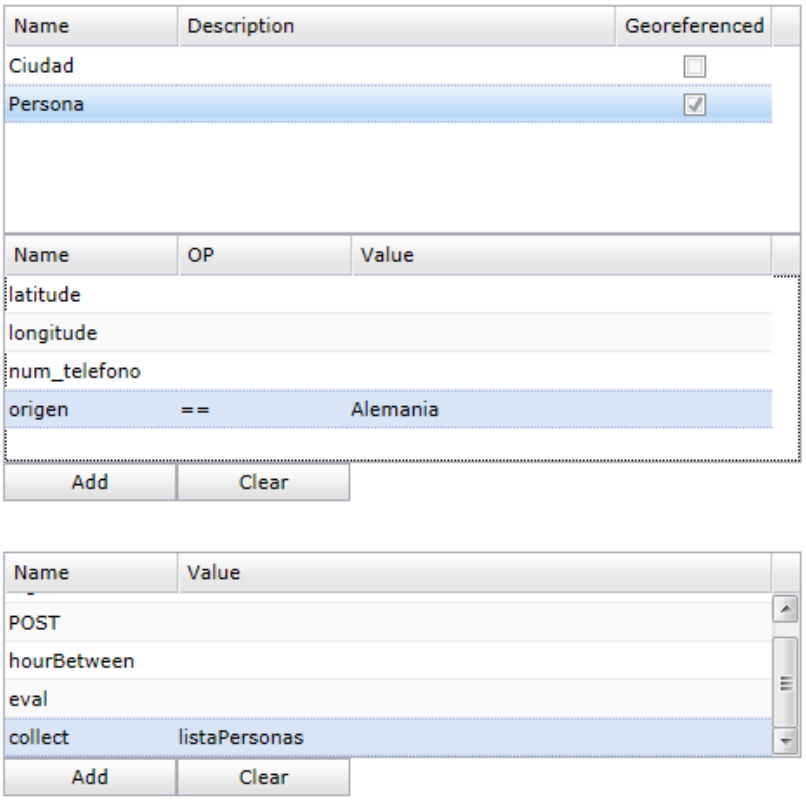


Fig. 87. Función *collect*

La configuración previa recolecta aquellas entidades de tipo "Persona" que se encuentren en la Base de Conocimiento y que cumplan la condición de tener la propiedad origen igual a Alemania. Esta configuración genera la siguiente condición en la regla.

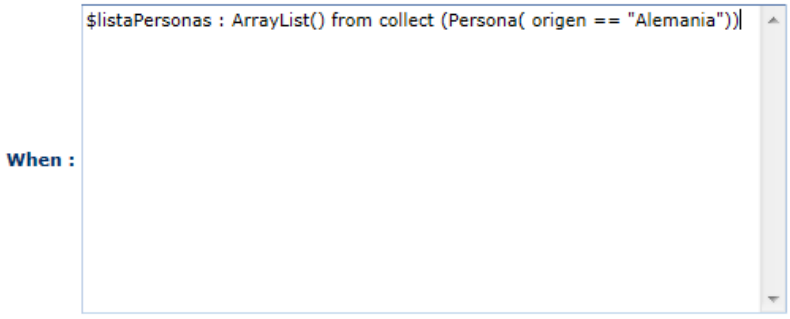


Fig. 88. Resultado función *collect*

- *accumulate*: recolecta entidades que cumplen un cierto filtro establecido sobre una lista, acumulando algunos valores de las propiedades de las entidades recolectadas. Los operadores posibles son: *average, min, max, count, sum*. Se utiliza como apoyo la entidad *Number*[53], que representa a un número con valores de diferentes tipos. A continuación se muestran dos ejemplos. En el primero se acumula en la variable "$total" la suma de los precios de las entidades de tipo "Producto" que se encuentren en la Base de Conocimiento y que pertenezcan a un pedido determinado, siempre y cuando sea superior a 100.

```
35. $pedido : Pedido(id="2")
36. //Acumula el precio en una variable de tipo Number
37. $total : Number( intValue > 100 )
38.              from accumulate( Producto( pedido == $pedido, $precio: precio ),
39.                               sum( $precio))
```

En este segundo ejemplo se acumula el número de áreas de tipo "Museo" en las que se encontraba una entidad de tipo "Persona".

```
40. $p : Persona()
41. //Acumula el número de áreas visitadas de tipo Museo
42. $total : Number() from accumulate($a : Area( type == "Museo" ) from $p.pastAreas,
    count($a))
```

Operaciones sobre entidades

En la consecuencia de una regla, pueden modificarse las entidades que se utilizan en las condiciones de la propia regla. En el ejemplo que sigue a continuación, en la línea 43 se aprecia el consecuente de una regla donde se modifica la edad de la persona mediante el acceso a dicha propiedad, proporcionado por la variable que almacena la entidad de tipo "Persona".

```
43. $persona.edad = 20;
```

También se pueden borrar entidades en el consecuente de la regla mediante la cláusula *retract*, tal y como muestra el ejemplo.

```
44. retract($persona)
```

[53] http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Number.html Último acceso 12-04-1982.

Además, se pueden insertar nuevas instancias de entidades mediante la cláusula *insert*. Esta cláusula recibe como parámetro la creación de una nueva instancia de una entidad determinada, teniendo como argumentos en su constructor las propiedades de la entidad (a excepción de las propiedades *latitude* y *longitude* en caso de ser geo-referenciada). Si por ejemplo se quiere crear una nueva entidad de tipo "Persona" como la que se ha utilizado en los ejemplos de secciones anteriores, los argumentos a especificar son *num_telefono* y *origen*, es decir, que hay que especificarlas en el orden en el que aparecen en el detalle de la entidad.

```
45. insert(new Persona(660000000, "España");
```

Además, se permiten las unificaciones entre diferentes entidades en base al identificador de cada una de las mismas. En la línea 46 se hace referencia a la instancia de la entidad de tipo "Persona" cuyo nombre es "David". Esta entidad contiene una propiedad denominada "cocheid", que almacena el identificador del coche del que es propietario el individuo. Posteriormente, en la línea 47 se materializa dicha relación mediante el operador "==", configurando la condición que unifica la instancia de entidad "Coche" con su propietario.

```
46. $persona : Persona(nombre == "David")
47. $coche: Coche( id == $persona.cocheid)
```

Operaciones temporales

Así mismo, se pueden realizar operaciones temporales con las instancias que están almacenadas en la Base de Conocimiento en la plataforma, mediante diferentes operadores temporales, como por ejemplo *after* y *before*. En el ejemplo que sigue a continuación, se pone como condición que la instancia de entidad "Persona" almacenada en la variable *$personaB* tiene que haber sido insertada en la plataforma después de *$personaA*.

```
48. $personaA : Persona()
49. $personaB: Persona( this after $personaA)
```

Anexo III. Manual del simulador *Siafu*

La herramienta *Siafu* está pensada para crear simulaciones de personas en movilidad (agentes), tanto a pie como en distintos medios de locomoción. De esta manera, se pueden recrear diferentes situaciones de las personas de una manera sencilla. En el simulador se ha representado concretamente una persona en movilidad por la ciudad de Donostia-San Sebastián. Los datos de contexto de la simulación serán los que se envíen a la plataforma *Context Cloud* con el fin de poder gestionarla e identificar situaciones.

Configuración

En la carpeta del simulador, se pueden encontrar diferentes archivos. Los más importantes son los siguientes.

- *runWin32.bat/runWin64.bat*: son los archivos con los que se ejecuta el simulador, dependiendo de si se ejecuta en una máquina de 32 o 64 bits.
- *simulador.properties*: fichero de configuración donde se especifican las propiedades para que el simulador pueda enviar datos de las simulaciones a la plataforma.
    - *domain* = dirección IP o dominio donde se encuentra la plataforma.
    - *port* = puerto donde se encuentra la plataforma.
    - *cloud_id* = identificador de la nube de contexto del usuario.
    - *provider_name* = nombre del proveedor pasivo que deberá estar creado en la plataforma y al que se enviarán los datos de contexto por parte del simulador.
    - *developer_key* = clave de desarrollador.

Controles

Al ejecutar el simulador se visualiza la siguiente pantalla. En ella se puede ver un mapa de la ciudad de Donostia-San Sebastián, una persona en color amarillo, y a la derecha un panel con diversa información.

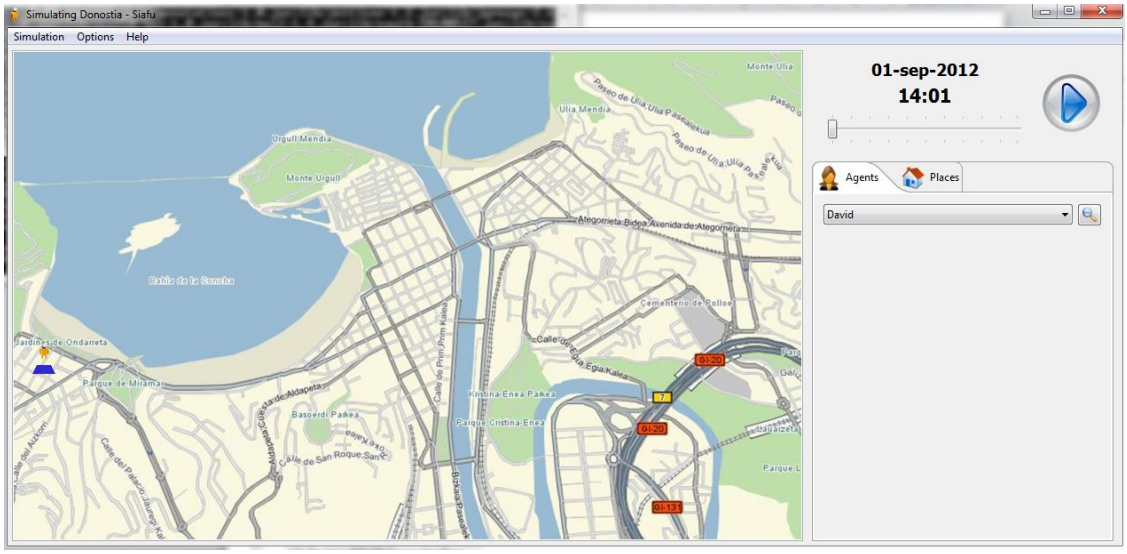


Fig. 89 Pantalla principal del simulador *Siafu*

Por un lado, se tiene la fecha y la hora del simulador **01-sep-2012 14:01**. Si se pulsa sobre el botón  se inicia la simulación, que puede pausarse cuando se quiera. Al iniciar la simulación, la hora del simulador empieza a avanzar. Con el control  se aumenta o disminuye la velocidad a la que avanza el tiempo en el simulador.

A su vez, pueden verse diversos detalles, tanto de la persona en movilidad como de los lugares en los que está. Pulsando en la pestaña de nombre "Agentes" *("Agents")* y dándole al botón  se pueden ver los detalles del agente, tal y como se muestra en la siguiente figura.

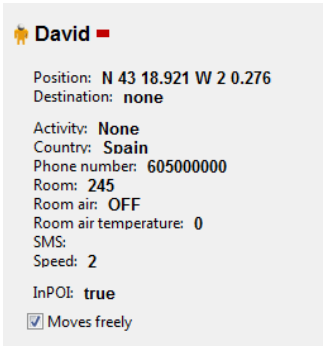


Fig. 90 Datos del agente

En el detalle del agente, se pueden ver diferentes datos de contexto.

- *Actividad (Activity)*: actividad de referencia del agente.
- *País (Country)*: origen del agente.

- ***Número de teléfono (Phone number)*:** número de teléfono del agente.
- *Habitación (Room)*: número de habitación de Hotel en el que se aloja el agente.
- *Aire habitación (Room air)*: estado (ON/OFF) del aire acondicionado de la habitación del agente. Se puede interactuar con este parámetro mediante el API definida que se describe en la siguiente sección.
- *Temperatura del aire de la habitación (Room air temperature)*: temperatura a la que está el aire acondicionado. Se puede interactuar con este parámetro mediante el API definida que se describe en la siguiente sección.
- *SMS*: en este campo se visualizarán los mensajes que se envíen al simulador. Existe un API en el simulador que se describe en la siguiente sección, con el que se puede interactuar para enviar mensajes al agente y que se visualicen en este campo.
- ***Velocidad (Speed)*:** velocidad a la que se mueve el agente.

Los datos señalados en negrita, son los que pueden ser enviados a la plataforma *Context Cloud* mediante la configuración del simulador. Además, se envían por defecto los siguientes datos.

- Latitud
- Longitud
- Nombre del agente móvil
- Hora del dispositivo móvil (igual a la del simulador)

Así mismo, pulsando en la pestaña "POI" se pueden ver los cuatro puntos de interés que se han definido sobre el mapa.

- *Hotel:* hotel donde se aloja el agente.
- *Bus Stop 1*: parada de bus.
- *Bus Stop 2*: parada de bus.
- *Gros Beach*: playa de Gros.

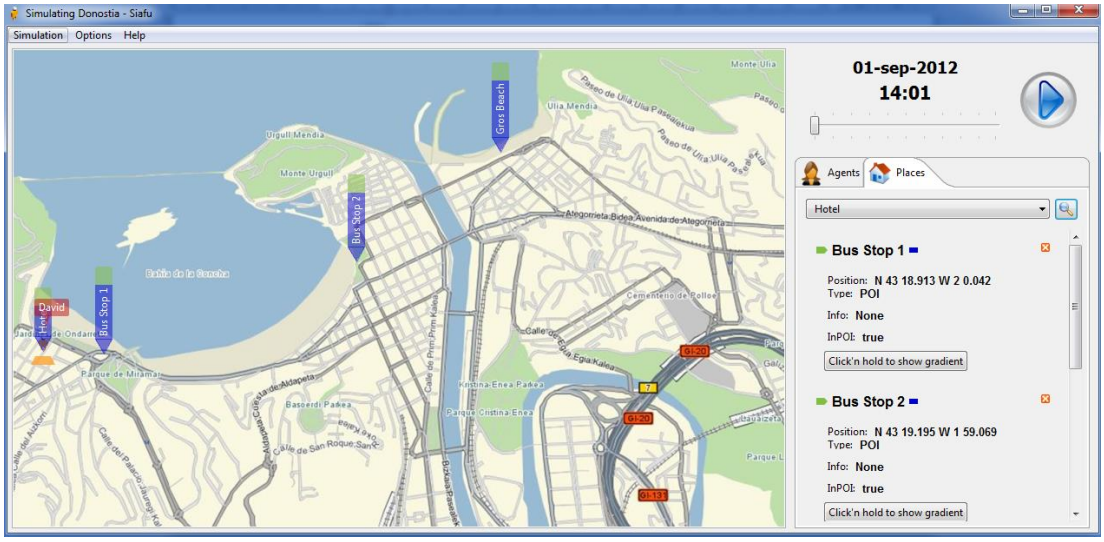


Fig. 91 Situación de los puntos de interés definidos para la simulación

API del simulador

El simulador ofrece diferentes servicios en *localhost* donde poder consultar diversa información de contexto del agente y de diferentes elementos de la simulación. Esta información es de utilidad para que el programador pueda configurar los proveedores necesarios con el fin de obtener la información de contexto pertinente. A continuación se describe cada uno de ellos, así como la forma de acceso a los mismos.

- *Servicio web meteorológico:* proporciona la temperatura para la ciudad de Donostia.

  (GET) http://localhost:8082/simulator/sources/weather

- *Detector de presencia de la habitación del hotel:* devuelve el número de personas detectadas en la estancia. Para ello hay que llamar al siguiente servicio indicando el número de habitación como parámetro {*roomnumber*}.

  (GET) http://localhost:8082/simulator/sources/room/{roomnumber}/presence

Anexo IV. Cuestionario de evaluación

Este cuestionario pretende saber tu opinión sobre diferentes aspectos relacionados con la utilización de la plataforma *Context Cloud*. Toda la información facilitada se tratará con absoluta confidencialidad.

¡Gracias por tu colaboración!

DATOS PERSONALES

Sexo: Masculino ☐ Femenino ☐

Edad: ☐ 20-25; ☐ 26-30; ☐ 31-35; ☐ 36-40; ☐ 41-45; ☐ 45-50; ☐ 50+

Formación académica

- ☐ Sin titulación universitaria
- ☐ Titulado universitario
- ☐ Master universitario
- ☐ Ph.D.

Profesión relacionada con las Tecnologías de la Información y Comunicación Sí ☐ No ☐

Conocimientos de programación software Nada ☐ Bajo ☐ Medio ☐ Alto ☐

Conocimientos de informática a nivel usuario Sí ☐ No ☐

CUESTIONARIO

A continuación se presentan una serie de afirmaciones. Señala por favor, qué grado de acuerdo o desacuerdo tienes respecto a las mismas según la siguiente escala.

1	2	3	4	5	6
En total desacuerdo	Muy en desacuerdo	En desacuerdo	De acuerdo	Muy de acuerdo	Totalmente de acuerdo

	¿Ha sido fácil de usar?	**1**	**2**	**3**	**4**	**5**	**6**
1	Aprender a usar el sistema ha sido fácil para mí						
2	Aprender la metodología de desarrollo ha sido fácil para mí						
3	He podido hacer con el sistema lo que quería en cada momento						
4	Es un sistema fácil de manejar y de interactuar con él						
5	Sería fácil llegar a ser un usuario experto en el manejo del sistema						
6	El sistema facilita el trabajo colaborativo						
7	La metodología de desarrollo facilita el trabajo colaborativo						
8	En general, tanto la metodología como el sistema son sencillos de usar						

	¿Ha sido útil?	**1**	**2**	**3**	**4**	**5**	**6**
9	Utilizar el sistema me ayudaría a realizar más rápido mi trabajo en el desarrollo de sistemas sensibles al contexto						
10	Utilizar el sistema mejoraría el rendimiento en mi trabajo en el desarrollo de sistemas sensibles al contexto						
11	Utilizar el sistema mejoraría la productividad en mi trabajo en el desarrollo de sistemas sensibles al contexto						
12	Utilizar el sistema aumentaría la efectividad en mi trabajo en el desarrollo de sistemas sensibles al contexto						
13	Utilizar el sistema facilitaría mi trabajo en el desarrollo de sistemas sensibles al contexto						
14	Es un sistema útil para mi trabajo						
15	El sistema es útil para realizar un trabajo colaborativo						
16	La metodología de desarrollo es útil para trabajar con el sistema						
17	La metodología de desarrollo es útil para realizar un trabajo colaborativo						
18	La metodología colaborativa es útil para implementar sistemas sensibles al contexto						

		1	**2**	**3**	**4**	**5**	**6**
19	Recomendaría el sistema a otros usuarios						
20	Utilizaría el sistema en futuros desarrollos de software						

Pagarías por utilizar el sistema: ☐ Sí ☐ No

¿Cuánto?_____€

21 Comentarios

This part of the evaluation will assess the factors associated with user acceptance of the *Context Cloud* platform.

Thank you!

PERSONAL INFORMATION

Gender: Male ☐ Female ☐

Age range: ☐ 20-25; ☐ 26-30; ☐ 31-35; ☐ 36-40; ☐ 41-45; ☐ 45-50; ☐ 50+

Educational background

- ☐ Without university degree
- ☐ University degree
- ☐ Postgraduate
- ☐ Ph.D.

Information and Communications Technologies (ICT) related occupation Yes ☐ No ☐

Programming skills No ☐ Low ☐ Medium ☐ High ☐

User level computer knowledge Yes ☐ No ☐

QUESTIONNAIRE

This questionnaire presents several statements. Please, use the following scale in order to evaluate each of the statements.

Unlikely	1	2	3	4	5	6	Likely
	Extremely	Quite	Slightly	Slightly	Quite	Extremely	

	Ease of Use	**1**	**2**	**3**	**4**	**5**	**6**
1	Learning how to use Context Cloud was easy						
2	Learning the methodology was easy						
3	I found it easy to get Context Cloud to do what I wanted to do						
4	The interaction with Context Cloud was clear and understandable						
5	It would be easy for me to become skilful at using Context Cloud						
6	The system eases the collaborative work						
7	The methodology eases the collaborative work						
8	The methodology and the system are easy to use						

	Usefulness	**1**	**2**	**3**	**4**	**5**	**6**
9	Using Context Cloud in my job would enable me to develop context-aware systems more quickly						
10	Using Context Cloud would improve my job performance in the development of context-aware systems						
11	Using Context Cloud in my job would increase my productivity in the development of context-aware systems						
12	Using Context Cloud would enhance my effectiveness in the development of context-aware systems						
13	Using Context Cloud would make it easier to develop context-aware systems						
14	I would find Context Cloud useful in my job						
15	Context Cloud is useful to do a collaborative work						
16	The methodology is useful to work with Context Cloud						
17	The methodology is useful to do a collaborative work						
18	The methodology is useful to develop context-aware systems						

		1	**2**	**3**	**4**	**5**	**6**
19	I would recommend other users to use Context Cloud						
20	I would use Context Cloud in future developments						

Would you pay for the system? ☐ Yes ☐ No
¿How much?_____€

21 Comments