proceedings

MDPI

Proceeding

A Domain Specific Language for Smart Cities †

Francisca Rosique [1]*, Fernando Losilla [1], Juan Ángel Pastor [1]

[1] Universidad Politécnica de Cartagena (UPCT), Cartagena 30202, Spain; Paqui.rosique@upct.es, Fernando.losilla@upct.es, juanangel.pastor@upct.es

* Correspondence: paqui.rosique@upct.es; Tel.: +34-96832-6589

† Presented at the 4th International Electronic Conference on Sensors and Applications (ECSA 2017), 15–30 November 2017; Available online: https://sciforum.net/conference/ecsa-4.

Published: 14 November 2017

Abstract: In Smart Cities Systems, one of the main problems of its development is the integration of different devices and sensors. Many of them are incompatible with each other (different platforms, communication protocols, etc.).In this paper a Domain Specific Language (DSL) for Smart Cities is presented. This DSL allow to describe the system of a Smart City using domain concepts, without considering platforms, protocols, etc. This DSL has been developed using a Model Driven Approach (MDA). Has therefore been developed a metamodel, adding additional constraints to validate the language and a textual and graphical syntax to enable the modeling of the Smart City System in an easy and simple way.

Keywords: Domain Specific Language; Model Driven Approach; Smart Cities

1. Introduction

The rapid advance in information technology, electronics and telecommunications has led to a broad integration of Information and Communication Technologies (ICT) in all sectors of society. These changes have opened the door to the digital world, the Internet of Things (IoT) and smart environments, which has allowed society to change the way in which it experiences and perceives the world. An increasingly digitized physical world, with sensors and digital instruments, as well as control devices, which aims at exploiting the most advanced communication technologies to support added-value services for the administration of the city and for the citizens.

Consequently, smart city systems constitute environments where the technological elements are invisible to the users but their functionality continues to be provided. Smart devices are inserted into daily tasks, making user-system interaction natural and uninhibited in all kinds of situations and circumstances [1]. Therefore, this concept does not only belong to the hardware area, but also to the software, constructing simple and easy interfaces that allow to focus the user's attention in the accomplishment of the task and not in how to carry it out.

In this complex scenario, the main challenge of software development for smart cities arises when integrating in the same system heterogeneous devices, with different protocols of communication. The Model Driven Software Development (MDSD) approach [2] seems to be a solution that is taking increasing importance in Software Engineering as a solution to these problems. The abstraction separation feature at different levels allows any system to be modeled independently of the platform, increasing dynamism and reuse, making it an ideal candidate for the integration of heterogeneous devices. In this paper, we present a specific language for smart city systems, based on the Model Driven Development approach. This DSL is textual and graphic, which facilitates the user's use and understanding.

The rest of this paper is organized as follows, in section 2, a specific language for the development of smart city systems using the Model Driven Development approach. Section 3

demonstrates its use in the case study of a mobility management system in a smart city. And finally in section 4 the conclusions will be presented.

2. Smart City DSL Metamodel

Smart City DSL is developed in Model Driven Engineering (MDE) and allows to model Smart City Systems using concepts close to the domain of the problem. Thus, the expert in these systems does not need to have additional knowledge in software engineering and the task of specifying the system will be much easier and intuitive.

As in any other language, whether general purpose or domain specific, Smart City DSL is composed of three fundamental parts for its development [3]:

- Abstract Syntax: Describes the vocabulary of concepts provided by the language and how they can be combined to create models. That is, it consists of a division of concepts, the relationships that exist between them and rules that establish how concepts can be correctly combined. Abstract syntax models are described in a metamodel language. For the case of Smart City DSL, which follows a Model Driven Approach (MDA), we use EMF (Eclipse Modeling Framework) [4].
- Concrete syntax: it is the notation that facilitates the presentation and construction of models and programs in the language. The specific syntax chosen for Smart City DSL is a textual and graphic syntax, which represents the system in the form of icon-based diagrams, allowing to express many details in a simple and intuitive way.
- Semantics: The semantics of a language describes the meaning of the concepts it handles and allows you to understand how to use it.
  Below is the syntax and semantics developed for Smart City DSL.

2.1. Abstract Syntax: Smart City Metamodel

The abstract syntax is provided by the developed metamodel (see Figure 1).

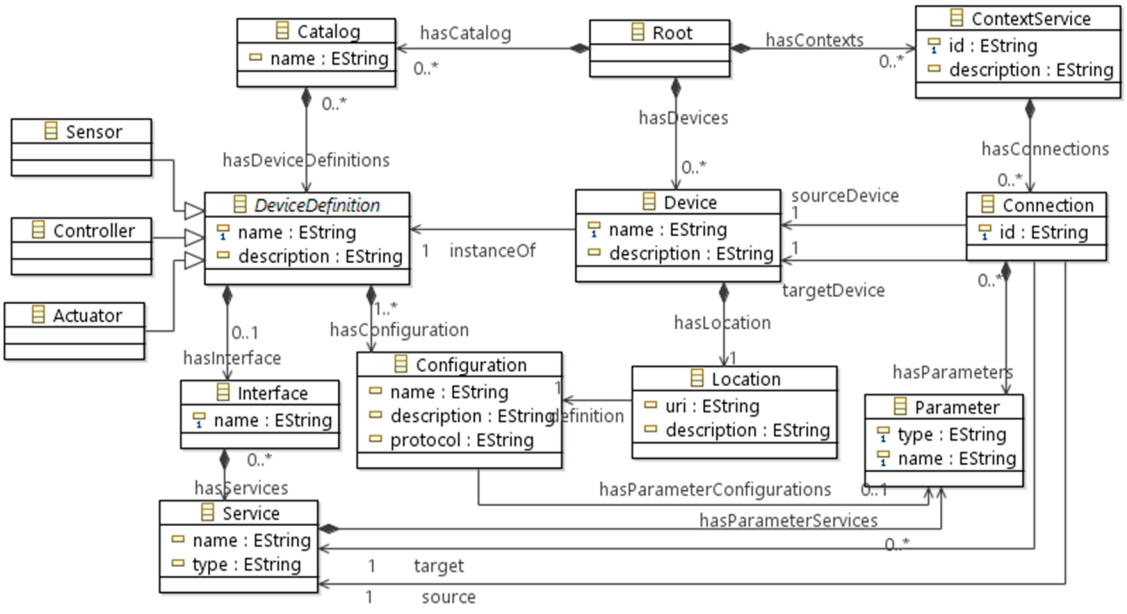

Figure 1. Smart City DSL metamodel.

The main element of this metamodel is *"Device"*. A *Device* represents a physical device of the system (a sensor, a router, a mobile, etc.) and is characterized by a unique identifier (name), its description and optionally an icon. Any device instantiated in an application will have a location (*Location*), so that the device will be located by its Uniform Resource Identifier (URI). The devices can connect and communicate with each other through the services (*Service*) offered and required by each of these devices. And depending on the context (*ContextService*) in which the system is integrated it

will be possible to establish connections between different devices without having to change the deployment of devices.

On the other hand, in the development of Smart City Systems it is usual to use devices that have already been defined in other systems or even have to include in the same system several identical devices. In order to promote the reuse of these devices and avoid having to define multiple times the same device for each application (even several times within the same application), it has chosen to use a catalog of device definitions (*DeviceDefinition*), which allows to define reusable devices, so that once defined this catalog can be used in any application and only need to instantiate devices (Devices). This catalog can also be edited to include, modify or delete devices and thus keep the catalog updated.

A *DeviceDefinition*, is the definition of a device, and can be classified into the *Sensor*, *Controller* or *Actuator* subtypes. In the definition of a device is indicated its interface, that is to say that services offers or requires such device, also indicates its configuration (*Configuration*), being the most important field of such configuration the protocol (*Protocol*). The protocol field will allow the system to be able to configure the lower layers of the architecture properly.

A fundamental part to achieve a precise and unambiguous specification of the models, both textual and graphics, are the additional restrictions on the objects of such models. These constraints are implemented in the form of rules or expressions that verify that the models satisfy the constraints defined by their metamodel, more specifically verify constraints associated with concrete metaclasses. To specify these constraints, OCL (Object Constraint Language) [5] has been used since it is being recommended by the OMG, especially in the context of Model Driven Engineering (MDE).

The main OCL restrictions implemented in Smart City DSL are shown below. The first OCL rules are responsible for preserving the uniqueness of those elements that can not be repeated in a model. For this it is verified that there are not two elements with the same attribute identifier. In the cases of the DeviceDefinition (See Rule 1) and Device (See Rule 2), this attribute is name. For localization it is the uri attribute (See rule 3), and for ContexService (see rule 4) the id attribute.

```
self.DeviceDefinition->forAll (n1,n2 | n1<>n2 implies n1.name <>n2.name)        (1)

self.Device->forAll (d1,d2 | d1<>d2 implies d1.name <>d2.name)                  (2)

self.Location->forAll (l1,l2 | l1<>l2 implies l1.uri <>l2.uri)                  (3)

self.ContextService->forAll(cs1,cs2 | cs1<>cs2 implies cs1.id <>cs2.id)         (4)
```

The following important rule (See Rule 5) affects connections between devices (*Connection*). The devices that are involved in a connection must be two different devices and the services that come into play must be complementary between them (they can not be of the same type, for example, can not be two services required).

```
self.Connectio+n->forAll ( c| c.sourceDevice.name != c.targetDevice.name
and c.source.type != c.target.type)                                             (5)
```

2.2. Concrete Syntax

For the definition of the concrete syntax, we have chosen the combination of a textual and graphic representation of the concepts defined in the metamodel.

For the definition of the device catalog model (DeviceDefinition) we have chosen the TreeEditor editor provided by EMF (Eclipse Modeling Framework) [4] that uses the persistent XMI (XML Metadata Interchange) format [6] to store the models. In the following figure (Figure 2.a) you can see a small extract from the catalog.

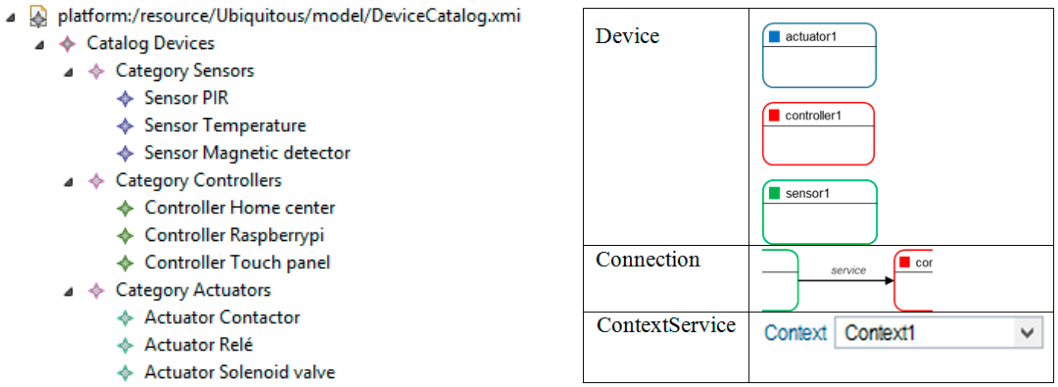

Figure 2. Concrete syntax of Smart City DSL. (**a**) extract of the device catalog; (**b**) graphic representation of main elements.

As can be seen the D*evices* are represented with a rounded rectangle with colored borders in: (1) blue if the device is of the *Actuator* subtype, (2) red for the *Controller* subtype, and (3) green for the subtype *Sensor*. Inside the rectangle an icon is displayed along with the name of the device. The icon can be an image selected by the user or the default icon (a square of the same color as the border). Connections between devices are represented by an arrow originating from the source device and destined for the source device. This arrow will have a tag that will indicate on behalf of the service involved in that connection. Finally the context (*ContextService*) is represented as a drop-down list accessible from the properties of the application itself. Depending on the selected context, an interconnection or other device will be displayed. So a device can be involved in different contexts at a time, but it can separate its functionalities and services in each occasion. Also thanks to this separation by context it is possible that the same application can be readapted simply by changing the defined context.

2.3. Semantics

Although Smart City DSL already employs concepts with their own meaning in the domain in which they are used (sensors, bulbs, etc.), a first definition of semantics is also provided through device descriptions. These descriptions are provided thanks to the device definitions *catalog*, which includes parameters, configurations, interfaces, etc., of each *DeviceDefinition*. This semantics will be completed more formally at lower abstraction levels of the MDA approach, for example at a lower level of component modeling. This new lower model would be obtained by carrying out a model-to-model transformation.

3. Case Study

To illustrate the use of the DSL for Smart City Systems presented, it is proposed to model a focused on the efficient management of citizen mobility. This system will be able to manage congestion situations of citizen mobility due to traffic jams, agglomerations in areas of special interest (hard to reach places, beaches, large areas, etc.) or events (demonstrations, sports events, concerts, etc.). .), as well as the observation in the first instance of emergencies or alerts. To this end, it will be necessary to integrate the use of services available in the "cloud" (mass storage, map generation, navigation, image processing, etc.), control and management of mobile road safety elements located at use of drone vehicles), and control of the operation of intelligent traffic lights, light poster and video surveillance cameras.

A modular, scalable, flexible and robust solution is proposed for the integration of distributed information in different parts of the city, allowing a growth in both the number of data sources and the volume of information from the sources. The different elements of the system should be able to make simple decisions to achieve the objectives, to interpret and react according to the environment, fulfilling the established requirements.

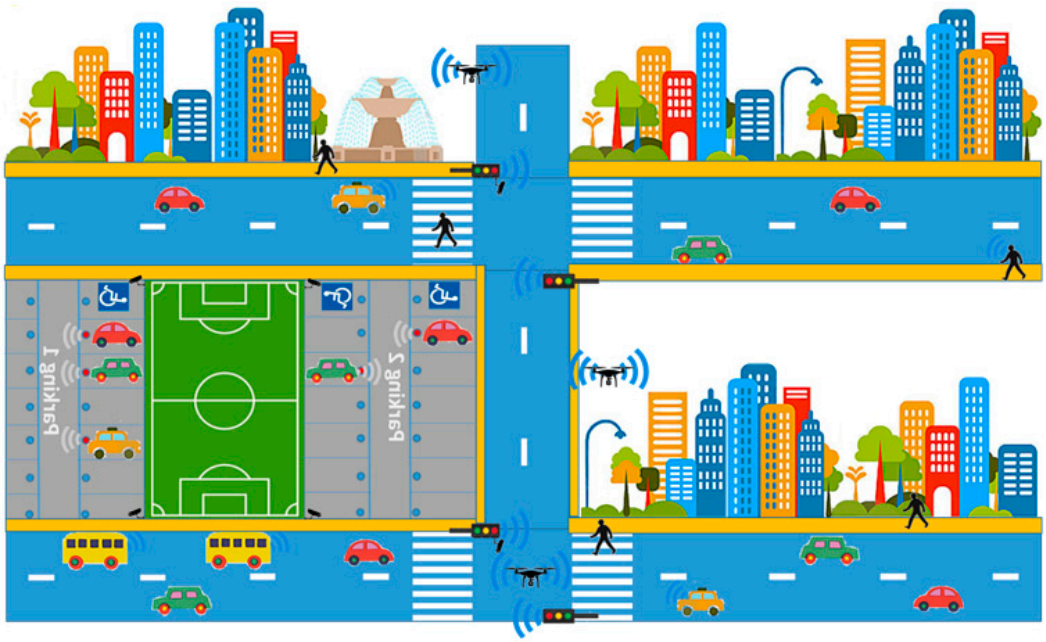

Figure 3. Citizen mobility management system.

The system shown in Figure 3 consists of the following devices:

- Network of Intelligent traffic lights equipped with sensors or cameras and 3G connection [7].
- Float of drones equipped with sensors, high resolution cameras, GPS and 3G connection. [8]
- Parking sensors. [9]
- Mobile devices (Smartphones, tablets, etc.).
- Cloud computing system. [10]
- Smart public transport (taxis and buses).
- IP surveillance cameras.
- Web application / app.

For this case study the following contexts have been considered according to the services offered: (1) Context of urban traffic management, (2) Context of management of interurban traffic, (3) Context of public transport management, (4) Context management of parking areas. Given the extensive modeling of the entire case study, it will be shown (see Figure 4) only the result obtained by modeling one of these contexts using the DSL language editor that has been developed.

3.1. Modeling the Context "Urban Traffic Management"

In the modeling of this first context, the following devices are involved: (1) Fixed or mobile "cameras" located at traffic lights, drones and in different locations of street furniture (correspond to the sensor type devices Camera1 and Camera2 represented in the fragment of the model shown in Figure 4). These devices collect information about the current state of the traffic and send it to the system controller (in this case the information is sent in video format). (2) Fixed or mobile "pollution sensors" located in traffic lights and drones (correspond to the device type SPollution1, SPollution2 sensor of the model fragment shown in Figure 4). (3) Cloud Computing system (corresponds to the "CloudComputing1" controller device of the model shown in Figure 4) (i) manages large volumes of information to be treated. (ii) Homogenizes the information between the different devices. Given the diversity of media (wireless networks, fixed networks, mobile networks), as well as the different formats, topologies and existing architectures that hinder the transmission of information between the different agents involved, it is necessary to homogenize the information. (iii) Perform control of the system. It handles the intelligent treatment of information and distribution of information for the correct operation of the Smart City system. (4) Finally, intelligent traffic lights (correspond to the "Traffic light" actuator type devices of the model shown in Figure 4) will receive the corresponding order.

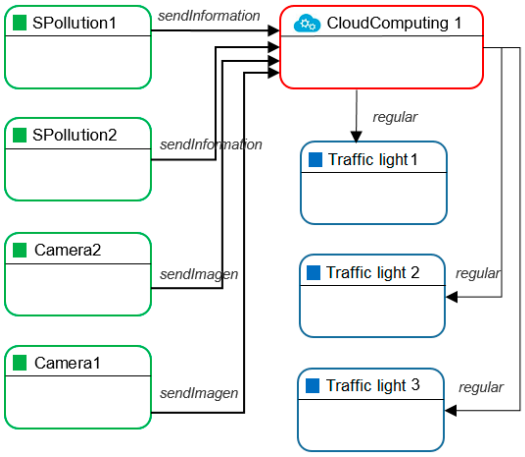

Figure 4. Fragment of the system model.

4. Conclusions

Nowadays, the development of ubiquitous systems (or pervasive systems) is a field of research at its peak. Traditionally, low-level abstraction technologies and specific frameworks have been used to develop such systems. In this article we have presented a specific language of the ubiquitous domain that allows to increase the level of abstraction and to facilitate the modeling of the ubiquitous systems. For this, a metamodel and a textual / graphic editor have been developed using the MDE development approach.

Thanks to the approach presented, a ubiquitous system can be specified using conceptual primitives suitable for this type of system. In addition, this approach can be supplemented with the lower levels of abstraction proposed by the model-led development approach (platform-independent level and platform-specific level) to generate functionally operable systems from the Smart City DSL-modeled specification.

References

1. Garfield, M.J. Acceptance of ubiquitous computing. Information Systems Management, 22(4): 24-31, 2005.
2. Selic, B. "The Pragmatics of Model-Driven Development," IEEE Software, vol. 20, no. 5, 2003, pp. 46–51.
3. Clark, T., Sammut, P., Willans, J. "Applied Metamodeling. A fundation for Language Driven Development (Third Edition)". 2015. arXiv:1505.00149.
4. Eclipse Modeling Framework (EMF). https://eclipse.org/modeling/emf/ (accessed on 2017).
5. Object Management Group: Object Constraint Language (OCL) Specification. http://www.omg.org/spec/OCL/ (accessed on 2017).
6. Object Management Group: XLM Metadata Interchange (XMI) Specification. http://www.omg.org/spec/XMI/ (accessed on 2017)
7. Albagul, A., Hrairi, M., Wahyudi, and Hidayathullah, M.F. Design and Development of Sensor Based Traffic Light System, American Journal of Applied Science, vol 3, no. 3, pp. 1745 – 1749, 2006.
8. Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., & Jawhar, I. (2014, May). UAVs for smart cities: Opportunities and challenges. International Conference on In Unmanned Aircraft Systems (ICUAS), (pp. 267-273). 2014.
9. Iam, Y., Kaiser, W., & Stathopoulous, T.. Implantar una red de sensores inalámbricos de NI para monitorizar la ocupación de plazas de aparcamiento. Revista española de electrónica, (663), 42-44, 2010.
10. Yamamoto, S., Matsumoto, S., & Nakamura, M.. Using cloud technologies for large-scale house data in smart city. In Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on (pp. 141-148). IEEE, 2012.